Advances in Session-Based and Session-Aware Recommendation

Dissertation

zur Erlangung des Grades eines

D o k t o r s d e r N a t u r w i s s e n s c h a f t e n

der Technischen Universität Dortmund
an der Fakultät für Informatik

von

Malte Ludewig

Dortmund

2020

Tag der mündlichen Prüfung: 12.11.2020

Dekan: Prof. Dr.-Ing. Gernot A. Fink

Gutachter:
Prof. Dr. Dietmar Jannach
Prof. Dr. Bernhard Steffen

Abstract

Personalized item suggestions provided by a recommender system have become a crucial part of many online services, e.g., shops or media streaming applications. Extensive evidence exists that such systems can improve the user experience and, at the same time, have a beneficial effect for the providers, e.g., in terms of the revenue. In academia, the recommendation problem is often framed as finding suitable items that users are not yet aware of based on their long-term preference profiles. In the real world, however, this problem formulation has a number of limitations.

Long-term profiles, for example, are not available for new or anonymous users and recommendations can then only be based on the few most recent interactions in an ongoing usage session. Various approaches to this relevant setting of session-based recommendation that recently gained more interest in the research community were proposed over the years. However, in terms of how to compare and evaluate such methods, no common standard has been established so far. The author of this thesis, therefore, developed an open framework for reproducible research and, furthermore, compared several approaches in a fair manner. Moreover, some of the techniques were proposed by the author himself. Extensive experiments and a user study, to some surprise, show that comparably simple nearest-neighbor techniques usually outperform recent deep learning models across many domains and datasets.

Even if long-term preference information is available about the users, recent works indicate that it might still be beneficial to consider the ongoing session, e.g., because a user visits a shop with a specific intent in mind. The author of this thesis, thus, conducted a systematic statistical analysis to assess what makes recommendations effective in what is called a session-aware scenario. This analysis is based on log data from a fashion retailer and the obtained insights were, furthermore, operationalized into novel session-aware recommendation approaches. Matching items of the customer's ongoing session, reminding him of previously inspected clothes, recommending discounted items, and considering recent trends in the community showed to be particularly effective strategies, not only for item recommendation but also in the related scenario of search personalization.

Contents

1 Introduction

Since the development of the World Wide Web by Tim Berners-Lee in the Swiss research facility CERN in 1989 and its introduction to the public in 1991, the network has grown rapidly. As of today, it has become a fundamental part of our lives and provides us with access to a nearly unlimited amount of information, entertainment, and services, e.g., news websites, media streaming platforms, and e-commerce shops. Google's video platform YouTube[1], e.g., by now is a host to hundreds of millions of videos with over 400 hours of video material being uploaded every minute [Bre15]. The music streaming service Spotify[2], for example, changed the way we listen to music and provides access to over fifty million songs and three billion playlists [Spo19]. As a consequence, it became impossible for users to fully explore all the options for entertainment, all the products they could potentially buy in e-commerce shops, or all the text they could read, e.g., on news platforms.

Hence, recommender systems became a necessary and even crucial part in most of today's online services, almost regardless of the domain and the application scenario. These systems try to understand the users in terms of their preferences and help them to discover additional items that they might also be interested in, e.g., new videos on YouTube, exciting tracks on Spotify, or a new shirt to buy on Zalando[3]. Besides these obvious examples, the application scenarios are almost endless. The social network Facebook[4], e.g., tries to recommend you the next event to attend or even new friends to make, the microblogging service Twitter[5] proposes people to follow, and the business network LinkedIn[6] suggests which job to apply for next.

In filtering the vast amount of information and presenting tailored item suggestions, undisputedly, recommender systems can have a beneficial effect for the user. This, in turn, can be measured in an increase of the perceived quality of the service in terms of the user experience, as, for example, shown in [CGT12; JJ19]. At the same time, the improved user experience not only helps the users of the service but on the other hand can also help the provider to increase, for example, the customer engagement and even the revenue [GH16; JJ19].

[1] https://www.youtube.com
[2] https://www.spotify.com
[3] https://www.zalando.com
[4] https://www.facebook.com
[5] https://www.twitter.com
[6] https://www.linkedin.com

Historically, the research area of Recommender Systems originates from the field of Information Retrieval and developed to its own field in the 1990s. With the famous Netflix Prize on movie recommendation [Kor09], the field enjoyed strongly growing interest. In general, two types of approaches dominated the publications, a) collaborative filtering techniques that try to find patterns in collective user behavior (see, e.g., [KBV09]), and b) content-based techniques that try to find similar items with the help of content information, e.g., the actors, the director, or the description of movies [MKP03]. In collaborative filtering, many publications framed the problem as a matrix completion task, where the matrix represents the preferences of all users towards all items and missing entries should be predicted. In the beginning, the proposed techniques often relied on matrices of explicit ratings (e.g., movie ratings from one to five) in order to learn a preference model and estimate the missing ratings. Later, the research focus shifted to representing the users based on implicit feedback, e.g., with the unary signal if the user clicked on a specific video or listened to a certain track.

By now, in academic research, a majority of the works still focuses on approaches that solely rely on such long-term preference models to select items to be suggested to the user. However, in many application scenarios of recommendation techniques this type of preference model is often not available for many of the users, for example, because they are not logged in to the service, or because they are visiting an e-commerce shop for the first time. Thus, when determining suitable recommendations for these users, systems have to rely on only a small amount of information, i.e., the users' few last interactions with a service or a website. Recommendation techniques that *solely* base their suggestions on an ongoing *session* and adapt to the users' actions in these are called *session-based* recommendation approaches.

However, even in cases where long-term information is available for a user, considering the current *session* might still contribute to the performance of a recommender system. In an e-commerce scenario, for example, the user might visit an online fashion store with a specific intent in mind. Assuming the user's focus lies on finding a new t-shirt, the recommendation system should immediately react to his needs, and not suggest, e.g., a winter jacket based on long-term preferences. Dependent on the season or changing trends, furthermore, long-term information might even not be valuable at all. Recommendation techniques that have the ability to *adapt to* an ongoing *session* and, at the same time, consider longer-term preferences are called *session-aware* recommendation approaches.

Both scenarios have particularities, challenges, and open problems, which have not been covered in research before. This thesis consists of a number of publications by the author of this thesis and addresses various of those, including, for example, the proposal and extensive comparison of session-based recommendation techniques,

the exploration of domain-specific aspects in session-based recommendation, or a systematic analysis on what makes recommendations successful in a session-aware e-commerce scenario.

The remainder of this chapter first provides a brief history of recommender systems in academia and explains the shifting focus from explicit to implicit feedback (Section 1.1). Subsequently, the differences between session-based, session-aware and the more general class of sequence-aware recommendation techniques are explained in more detail (Section 1.2). After briefly discussing evaluation schemes and quality measures for sequence-aware recommender systems (Section 1.3), the main research questions of this thesis are presented (Section 1.4). Finally, the overall structure of the remaining parts of the thesis is outlined (Section 1.5) and, furthermore, a list of the publications that are included is presented (Section 1.6).

1.1 From Explicit to Implicit Feedback

As previously mentioned, besides content-based approaches, personalized recommendations are often based on collective user feedback, which can either be available as *explicit* or *implicit* feedback. Over the years, the focus of recommender system research shifted from using *explicit* to considering *implicit* feedback. In the following, this shift is explained in a brief historic overview of popular recommendation techniques from both worlds. The session-based and session-aware approaches discussed in this thesis solely rely on different types of *implicit* feedback signals. Thus, this section additionally provides an overview of these.

Historically, the research in the field of recommender systems was dominated by scenarios, in which the users *explicitly* stated their interest in items, mostly in the form of item ratings on a numerical scale, e.g., from 1 to 5 [JWK14]. In consequence, many sophisticated algorithms have been proposed that were very accurately able to predict the rating that a user would probably give an item. As mentioned earlier, such approaches mostly frame the problem as a user-item rating matrix completion task and rely on large datasets of historical ratings.

Over the years, k-nearest-neighbor techniques (see, e.g., [Sar+01]) that often rely on finding similar users or items based on cosine similarities in the user-item rating matrix, were outperformed by matrix factorization techniques that represent users and items as lower-dimensional latent vectors. A very prominent example of such a technique is FunkSVD, which is based on the concept of singular value decomposition and, furthermore, played a major role in the winning solution to the 2009 Netflix Prize on rating prediction [Fun07; Kor09].

While in some domains plenty of explicit feedback is given by the users, e.g., for videos or restaurants, there are several practical applications, where only little or even no explicit feedback is available [JH09; JKG12]. In a business social network, for example, jobs or co-workers will hardly be rated and, thus, job or friend recommendations can solely be based on behavioral information monitored for the users, the *implicit* feedback.

Also in implicit feedback recommendation scenarios, matrix factorization techniques became a prominent choice to approach the problem. Here, instead of explicit ratings, the user-item matrix often contains unary, binary, or graded feedback. Very prominent examples of such techniques are Bayesian Personalized Ranking (BPR) that tries to learn a personalized ranking from a unary feedback matrix, and a model based on alternating least squares optimization (ALS) that included watching times of TV programs as implicit feedback [Ren+09; HKV08; TPT11].

In general, *implicit* user feedback can be divided into a) *directly observable user actions*, b) *feature-related indirect preference signals*, and c) *user-action-related indirect preference signals* [JLZ18]. Unary feedback like item click events as used in BPR can be assigned to the *directly observable user actions*, while the watching time incorporated by ALS can be classified as a *user-action-related indirect preference signal*. A *feature-related indirect preference signal* would, e.g., be the explicit statement of liking a particular brand, which can then indirectly by used in a content-based recommender system.

In the scenario of session-based or session-aware recommendation, respectively, the user feedback, in general, is naturally implicit, mainly in the form of *directly observable user actions*. The input is mostly formed by a stream of user-generated events, e.g., clicks on items in an e-commerce browsing session, or listening events in a mobile music streaming application. However, the collected signals can, of course, also include explicit rating, for example, thumbs-up/down statements on a video streaming platform for a specific clip.

Finally, according to [JLZ18] the collection of implicit feedback has some general problems or disadvantages that also session-based and session-aware recommendation techniques are naturally affected by. These, for example, include difficulties regarding the *interpretation of a signal*. When a user interacts with an item, it still remains unclear if the detailed inspection leaves the user with a positive or negative impression. The signal can only be interpreted as positive feedback or discarded based on some criterion, e.g., that the item was only inspected for a very short time, which might indicate a negative impression. In general, however, implicit feedback systems have a *lack of negative signals*. Furthermore, it is hard to assess the *strength of the signals*. Finally, like rating datasets, also implicitly collected feedback is prone

to *skewed distributions* in terms of the item popularity. Training recommendation models based on such feedback might lead to a popularity biased system that seldom recommends unpopular or niche items to the users.

1.2 Sequence-Aware, Session-Aware, and Session-Based Recommendation

Many traditional recommendation approaches often ignore important particularities of practical application scenarios. A majority of the techniques, for example, neglects the order of the user feedback for the creation of recommendation lists. Considering such information, however, might be crucial to present the user with valuable recommendations. In the following, a few important scenarios are described:

- Independent of the domain, a change in taste might be detected for users over time. They are subject to *long-term interest-drifts* and might, e.g., start to preferably listen to other musical genres or change personal styling choices due to a new job. Parts of the historical user feedback, thus, can become obsolete.
- Even when there is no change in the user's taste, other temporal dynamics might decrease the value of historic user feedback, e.g., *seasonal trends* in an e-commerce scenario.
- Although a user is well-known to an online retailer, he might visit a shop with a specific *short-term intent* in mind that a traditional recommender system cannot be aware of based on the user's long term interaction history.
- In the process of creating a playlist in an online music application, the *order of the clicks* in the historical user feedback might be of great importance. Certain tracks might often appear in the exact same order as they harmonically match each other, which a recommender system, in this particular scenario, should be aware of when recommending the next track to add.
- Traditional recommender systems focus on presenting items that users are not yet aware of. However, in an e-commerce scenario or in the music domain, for example, it might be useful to *remind* the user of already known items that he has not yet purchased or that he likes to listen to in certain situations.
- Finally, a user might be new to a service or simply not logged in. As a consequence, in this user *cold-start* setting personalized recommendations can, thus, solely be based on the very few first interactions with the platform.

Many similar patterns can be found throughout various practical application domains and, thus, led to new research areas within the field of recommender systems that emerged from the traditional matrix completion task, specifically, *context-aware*,

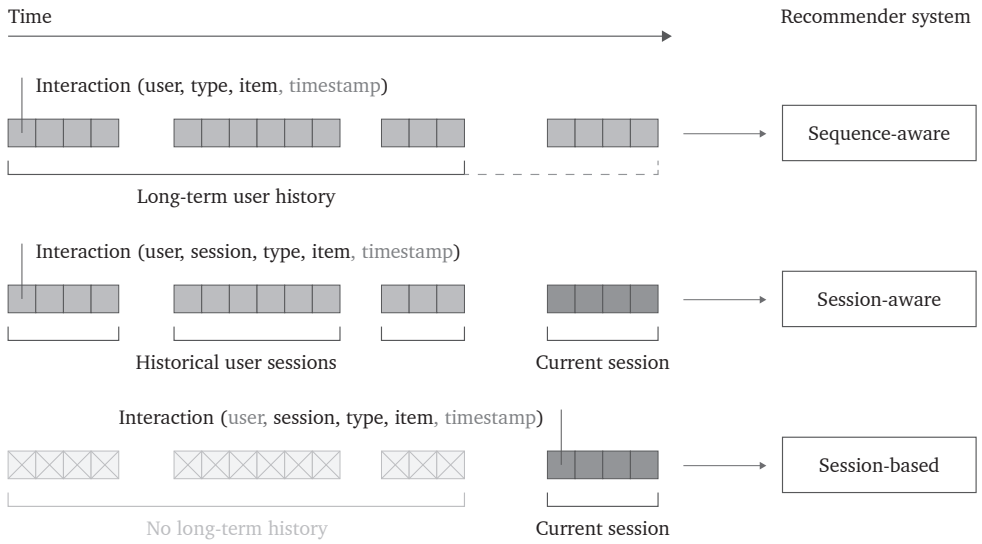


Figure 1.1: A visualization of the similarities and differences of the sequence-aware, session-aware, and session-based recommendation scenarios.

time-aware, and *sequence-aware* recommendation [QCJ18]. In the context of implicit feedback, all of these areas, however, share many common ideas and show similarities in terms of the input to a recommender system as well as its evaluation. The scenario of *sequence-aware* recommendation can be differentiated from the other areas in paying special attention to repetitive sequences in the users' feedback. Furthermore, the settings of *session-aware* and *session-based* recommendation are closely related to and can be classified as a subproblem of *sequence-aware* recommendation. Figure 1.1 visualizes the main inputs to all three problems and, thereby, demonstrates the key differences.

The input to a *sequence-aware* recommendation problem, in general, is a list of past user interactions in chronological order. In some cases, these interactions are timestamped, which then also allows the application of time-aware recommendation concepts. Each interaction usually refers to an item from the catalog and also to a predefined type of interaction, e.g., a click on an item or its purchase. Furthermore, items, as well as users, might have additional attributes. An item in an e-commerce shop, e.g., could be associated with a specific category or brand while the user has demographic attributes like his age. Due to the lack of publicly available datasets with additional metadata information and, at the same time, for the benefit of comparability and reproducibility, techniques presented in the literature often rely only on one interaction type and do not consider additional metadata.

In the more specific case of *session-aware* recommendation, besides a user identifier, each interaction is furthermore associated with a session identifier that groups several consecutive interactions. This type of information is very common in online services that are accessed by a web browser. Here, interactions are grouped into a

single session until the user does not perform any interaction for a specific amount of time (often configured to thirty minutes) or logs out. The additional information can be helpful in terms of identifying the previously mentioned *short-term intents*, as a user might start a new session with a specific goal in mind. In contrast to *sequence-aware* recommendation, *session-aware* recommenders necessarily always consider the users' current session. In *sequence-aware* recommendation, some early techniques rely only on the long-term history of users while still taking the order of events into account. Often, session-aware recommender systems are designed as a combination of long-term and short-term interest models, e.g., in an e-commerce setting or for mobile application recommendations [JLJ15a; Bae+15].

Session-based recommendation can be seen as both *sequence-aware* or *session-aware* in a user cold-start scenario, i.e., in the absence of historic information for all users. In contrast to the *session-aware* and *sequence-aware* problem, the *session-based* setting is particularly challenging, as no personalized long-term models can be used, and the personalization of item suggestions can solely be based on the last few user interactions. Typical application examples include news, e-commerce, video, and music recommendation [GDF13; Hid+16a; Hid+16b; HMB12].

Across all three scenarios and as also in traditional "item-ranking" recommender systems, the output of a sequence-aware approach is usually simply an ordered list of items[7]. The main computational tasks can be identified as a) Context Adaptation, b) Trend Detection, c) Repeated Recommendation, and d) Consideration of Order Constraints and Sequential Patterns (refer to [QCJ18] for a detailed discussion).

Over the years, a variety of different approaches has been proposed to generate such *sequence-aware* recommendations, both simple and very complex ones for all three scenarios. The presented methods range from comparably simple frequent pattern mining techniques [HLC08; LLC14], over adapted nearest-neighbor-based or matrix-factorization-based approaches [JLJ15a; Twa16], to the recent proposal of many neural network-based models [SEH16; Hid+16a].

This thesis focuses only on the specific settings of *session-based* and *session-aware* recommendation. A number of research questions regarding these scenarios that are covered by this work are presented in Section 1.4. Furthermore, relevant related work will be discussed extensively throughout the next chapters alongside a more formal definition of the explored scenarios.

[7] In some very specific scenarios, the recommended list is not a set of alternatives but a sequence that the user should consume in the exact order.

1.3 Evaluation in Session-Aware and Session-Based Recommendation

After differentiating the specific recommendation scenarios, it is furthermore important to understand how the quality of a recommender system can be assessed. This section, thus, briefly introduces possible ways that are widely used throughout the literature with a particular focus on session-based and session-aware recommendation. At the same time, some relevant problems are be identified.

In general, the evaluation of a recommender system can be divided into *offline* experimental schemes and *online* live experiments. In an *offline* setting, usually, parts of the user feedback are hidden from the recommender system, which should then most accurately predict the missing information. In an *online* experiment, in contrast, the system is deployed and the recommendations are presented to real users. For both scenarios, many different metrics exist in the literature that should capture the quality of the generated recommendation lists.

1.3.1 Offline Evaluation

As described in the previous section, the main input to session-aware or session-based problems is an ordered list of user events with session identifiers. Naturally, sessions should not be split, as they might represent interactions corresponding to a specific purpose, and also the order of the events should remain intact. Regarding *offline* evaluations, according to [QCJ18] the data is mainly partitioned into a training and a test part in the following two ways:

- For the scenario of session-based recommendation, long-term user histories do not exist. Hence, the common partitioning scheme is mainly *time-based* or *order-based* at the *session-level*. From the set of all available sessions, the most recent N sessions are used for testing the system. The number of sessions N can, e.g., be defined by a constant number, by a certain split date, or as a percentage of the whole dataset (often 10% or 20%).
- In session-aware recommendation, long-term user histories have to be considered. Here, the dataset is usually split *user-wise*, either for all users (*community-level* partitioning) or only a subset of all users (*user-level* partitioning). For each test user, again, a *time-based* partitioning at *session-level* is performed.

In consequence, the test data consists of a number of sessions, either with or without a long-term user history. In general, sequence-aware approaches try to predict the future actions of a user. Hence, the common idea of all evaluation schemes is to reveal some actions from the beginning of the test sessions and use the following

actions as the ground truth. These indicate what the system should recommend. The following two generic protocols can be identified from publications regarding session-based and session-aware recommendation:

- *Given-N next-item prediction.* In this protocol, the first N interactions of a test session are revealed to the recommender and only the immediate next interaction defines the *ground truth* for evaluation. In some works, only the last action of each sessions is hidden (see, e.g., [HMB12]). In other works, in contrast, the number of revealed interactions N is incrementally increased [Hid+16a].

- *Given-N remaining items prediction.* The second protocol is very similar to the first one. However, instead of defining only the immediate next action as the ground truth, all remaining hidden actions in a session are assumed to be relevant to the user. Besides incrementally increasing number N of revealed actions, again, N is sometimes static [JLJ15a].

In general, according to [QCJ18] a multitude of very specific evaluation schemes has been presented in the literature for both scenarios and no common standard can be found. Furthermore, evaluations in the literature have been performed on a variety of different datasets, which, overall, has counterproductive effects on the comparability of proposed techniques.

With regard to the *ground truth*, recommender systems are mainly evaluated in terms of *accuracy* metrics in an offline setting. Depending on the evaluation protocol, the following measures are employed in session-based and session-aware scenarios:

- For the task of *next-item prediction* when only one item forms the ground truth, the dominant metrics are the Hit rate (HR, Recall with only one relevant item) and the Mean Reciprocal Rank (MRR). While the HR only measures if an item is included in a recommendation list, the MRR also considers the rank in the list.

- When the ground truth consists of multiple items, standard measures from the field of information retrieval can be applied. These, e.g., include Precision, Recall, Mean Average Rank (MAR), Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG), and the F1 metric.

As mentioned in Section 1.1, the distribution of the feedback for items in datasets might by skewed towards a few very popular items. As a consequence, a recommender system that was trained on this data might be popularity biased, i.e., it often suggests these very popular items, because it is statistically beneficial in terms of accuracy metrics. In a real-world scenario, however, these recommendations might not be satisfactory to a user and, moreover, increase the skew of the distribution. Thus, it is advisable to also evaluate the recommender systems in terms of *beyond-accuracy* metrics. A few examples are given in the following:

- The level of a potential popularity bias can, e.g., be measured with the average *popularity* of the items in the produced recommendation lists.
- The metric *coverage* has a similar purpose and measures the number of different items that a system has recommended overall. A higher value, thus, might indicate that the number of more unpopular items among the recommendations is also higher.
- According to [Eks+14] recommendation lists are more appealing to the user, when they offer a diverse selection of options. In some works, therefore, an *inner list similarity* is calculated as a quality metric. The lower the similarity is, the more diverse are the recommendations. Often, the similarity is defined by the co-occurrences of items or additional metadata.
- To practically apply a recommendation technique, the scalability of the approach is important. The time required for model training as well as for the generation of a recommendation list should not exceed certain limits. Hence, the measurement of, e.g., *running times* and *memory consumption* is another interesting *beyond-accuracy* metric.

Many more examples can be found in the literature, e.g., in [Her+04; GDJ10]. However, techniques that have recently been proposed in the field of session-aware and session-based recommendation seldom consider such quality aspects.

1.3.2 User Studies and Field Tests

Online experiments that deploy recommender systems and collect the feedback of real users can further be classified into a) user studies, and b) field tests. While user studies often frame a certain situation and subsequently ask the user qualitative questions, field tests rely on a production-like environment and collect implicit feedback of a large number of users, e.g., in an A/B test of two candidate algorithms. Here, typical metrics are, for example, click-through or conversion rates [JJ19].

In the research literature, both types of online experiments often show a discrepancy to offline results and are, thus, highly relevant to the recommender system research community. In academia, however, the possibilities of conducting large-scale online studies are often limited. Thus, also for the field of session-based and session-aware recommendation the number of existing studies is very low. Some of the few examples can be found in [BOL09; Gar+14; Grb+15; MBR12; KJ17].

In [BOL09] and [KJ17], the authors compared a number of "playlisting" approaches that can also be seen as session-based techniques through a user study (a playlist here corresponds to a listening session). Garcin et al. [Gar+14] deployed some session-

based techniques in a field test on a news website. The results generally reveal strong performances of these session-based techniques compared to popularity-based baseline approaches. However, the results also differ strongly from a previously conducted offline experiment.

Overall, as for other recommendation problems, determining a good metric to predict the success of a recommendation technique from offline results is very challenging also in session-based and session-aware recommendation setting.

1.4 Research Questions

Referring to the previously introduced characteristics and problems of session-based and session-aware recommendation settings, this section will now elucidate the open research questions that are covered in this thesis and the included publications, respectively. The questions are divided into two blocks, one for each specific scenario. The questions that motivated the research works with a focus on session-based recommendation are the following:

- As mentioned in Section 1.3, many approaches proposed in the literature often rely on a customized or unique evaluation scheme. Even if the protocol is exactly the same, publications also differ strongly in many other regards, e.g., the considered datasets, the dataset preprocessing, or the baseline approaches employed for comparison. Thus, it is often difficult to understand the relative performance of recently proposed techniques. One main goal of this thesis, hence, is to propose a unified research framework that allows for a fair comparison of session-based techniques. Furthermore, an extensive experimental comparison has been performed to answer the question: How do recently proposed session-based recommendation approaches compare in a reproducible setup?
- With regard to the employed baseline approaches, many of the publications on session-based recommendation are missing out on simple but effective nearest-neighbor techniques that produced accurate recommendations in a similar setting [JLJ15a]. Another goal of this thesis, thus, is to adapt these to the scenario of session-based recommendation and answer the questions: How well do such simple techniques perform in comparison to modern deep learning solution? And, are they scalable enough for practical application?
- The predominant evaluation scheme in the scenario of session-based recommendation is the *given-N next-item prediction*, where the dataset is usually split *time-based* at *session-level* (see Section 1.3). As not only the immediate next item might be relevant to the user but rather all remaining items in a session, the following questions arise: How do session-based techniques perform in *given-N*

remaining items prediction or other alternative schemes? How do the results compare to the commonly employed *given-N next-item prediction* scheme? And most of all, which evaluation setup is a better indicator for the online success of a recommendation technique?

- Publications in the area of session-based recommendation often limit their experiments to a small number of datasets or a specific domain. However, the performance and suitability of an algorithm might be highly dependent on such factors. Therefore, further questions are: How do recently proposed algorithms compare across multiple domains? Can particularities be found that maybe indicate which algorithm should be applied in which domain?

- In general, the quality of a recommender system is often reduced to its accuracy results in an offline setting. As mentioned in Section 1.3, this pattern can also be found for session-based recommendation. However, also other measures exist that can be an indicator for the performance and applicability of a recommendation approach, e.g., the coverage, or the scalability in terms of training and prediction times. Most works neglect these factors, which is why the analysis of additional quality measures in session-based recommendation is another research question of this thesis. Specifically, are the approaches prone to potential popularity biases? Can they easily be applied in practical scenarios in terms of their computational complexity?

- Finally, the evaluation of modern session-based techniques, so far, was only performed in offline experimental settings. As shown in, e.g., [Gar+14; KJ17], offline accuracy results can give a misleading impression of a recommender system's online performance. Hence, another goal of this thesis is to compare session-based techniques in an online experiment to answer the following questions: How do the most promising techniques compare in a user study? How do the users perceive the quality of the recommendation? Can offline evaluation results in session-based recommendation provide a good indication for online performance and, if this is the case, which metric is the best indicator?

In contrast to the questions regarding the scenario of session-based recommendation, which is approached in a very universal way, for the setting of session-aware recommendation the research questions target very specific problems and particularities from the e-commerce domain:

- With the ideas of, e.g., *reminding*, *short-term intents*, and *seasonal trends*, Section 1.2 mentioned several interesting problems or aspects that contribute to the ongoing proposal of new session-aware recommendation techniques. In academic research, publicly available datasets usually consist of user interaction logs. Sometimes these interactions also include recommendation click events, but almost never the list of recommended items itself. Thus, it is hard to analyze

and quantify the real importance of such aspects. The author of this thesis, however, was provided with a non-public dataset that includes exactly this type of information for log data from the large European online fashion retailer Zalando. This opened the opportunity to extensively investigate, why users click on specific recommendations. In a structured analysis, this thesis, thus, addresses the following questions: What makes a recommendation successful in this particular e-commerce scenario? Why and when do users click on recommendations? How important are aspects like reminding, short-term intents, and seasonal trends in the fashion domain?

- Corresponding to the potential results of such an analysis, furthermore, the following questions arise: How could the obtained insights from the analysis be operationalized to build a more effective session-aware recommender system? How does such a system perform in an offline evaluation setup in terms of standard accuracy metrics?
- The session-aware recommendation scenario, at least in the e-commerce domain, shares many similarities with the related scenario of personalized search. The input to both problems similarly includes a stream of past user interactions. In personalized search, however, also a search query is provided. Assuming that the query solely narrows the set of potential items to recommend from, session-aware algorithms can directly be applied to the problem of search personalization. In a case study, this thesis addresses the following questions: Are session-aware recommendation techniques useful to personalize search results? How do they perform in comparison to techniques from the field of personalized search? Can the findings from the analysis regarding successful recommendations contribute to developing effective recommendation techniques in this scenario?

More details on how and why these questions have been developed and answered will be discussed throughout the following chapters.

1.5 Structure of the Thesis

The remainder of this thesis is organized as follows. Chapter 2 focuses on the research questions regarding the scenario of session-based recommendation. After the setting is formally defined, session-based recommendation approaches from the research literature are extensively reviewed. The approaches can be categorized into nearest-neighbor-based, frequent-pattern-based, factorization-based, and neural-network-based techniques. In the following, these techniques are compared in an extensive multi-dimensional offline evaluation. The comparison includesmany domains, datasets, evaluation protocols, and metrics, both in terms of accuracy measures and beyond-accuracy quality factors [JL17c; KJL17; LJ18b; Lud+19].

Next, the evaluation of session-based recommendation approaches in user studies and field tests is discussed in more detail. In this context, the approach from [LJ19b] to a user-centric evaluation of a selected number of session-based recommendation approaches is presented.

Chapter 3 focuses on two particular case studies in the scenario of session-aware recommendation that have been presented by the author of this thesis in [JL17a], [JL17b], and [JLL17]. First, a structured analysis of log data in the e-commerce domain that helps to identify the characteristics of successful recommendations is presented [JL17a]. The findings are, furthermore, utilized to present a novel approach to session-aware fashion recommendation [JLL17]. Finally, a number of selected session-aware recommendation techniques was tested in the related setting of e-commerce search personalization [JL17b].

Chapter 4 concludes the previous parts of this thesis and summarizes the main findings. Furthermore, open questions and future perspectives in session-based as well as session-aware recommendation are discussed. Finally, eight of the author's publications that lay the foundation for this thesis are included. A list of these can, furthermore, be found in the following section.

1.6 Publications

1.6.1 Covered Publications

This thesis is based on eight of the author's publications. The contribution to each publication is described in the following. Furthermore, a complete list of all publications as well as the publications themselves can be found in the appendix.

Determining Characteristics of Successful Recommendations from Log Data: A Case Study

Dietmar Jannach and Malte Ludewig. "Determining Characteristics of Successful Recommendations from Log Data: A Case Study". In: *Proceedings of the 32nd ACM SIGAPP Symposium On Applied Computing*. SAC '17. 2017, pp. 1643–1648

This publication is a joint effort with Dietmar Jannach. The author of this thesis developed the proposed framework for determining insights on what is making particular recommendations successful. Furthermore, he wrote parts of the text.

Investigating Personalized Search in E-Commerce

Dietmar Jannach and Malte Ludewig. "Investigating Personalized Search in E-Commerce". In: *Proceedings of the 30th International Florida Artificial Intelligence Research Society Conference*. FLAIRS '17. 2017, pp. 645–650

In a joint effort with Dietmar Jannach, this work investigates the performance of session-aware recommendation techniques for personalized search in an e-commerce scenario. The author of this thesis designed, implemented, and executed the evaluation as well as parts of the algorithms in the context of the reported experimentations. Moreover, he wrote parts of the text.

Session-Based Item Recommendation in E-Commerce: On Short-Term Intents, Reminders, Trends and Discounts

Dietmar Jannach, Malte Ludewig, and Lukas Lerche. "Session-Based Item Recommendation in E-Commerce: On Short-Term Intents, Reminders, Trends and Discounts". In: *User Modeling and User-Adapted Interaction* 27.3-5 (2017), pp. 351–392

This joint work with Dietmar Jannach and Lukas Lerche is an extension of the works presented in [LJL16] and [JL17a]. Besides writing parts of the text, the author of this thesis contributed by designing and evaluating model-based two-stage approaches to successfully operationalize previous findings regarding the characteristics of successful recommendations in e-commerce.

When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation

Dietmar Jannach and Malte Ludewig. "When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation". In: *Proceedings of the 11th ACM Conference on Recommender Systems*. RecSys '17. 2017, pp. 306–310

This publication is joint work with Dietmar Jannach. The author of this thesis wrote parts of the text, contributed to the design of experiments, executed them, and helped with the analysis of the results.

A Comparison of Frequent Pattern Techniques and a Deep Learning Method for Session-Based Recommendation

Iman Kamehkhosh, Dietmar Jannach, and Malte Ludewig. "A Comparison of Frequent Pattern Techniques and a Deep Learning Method for Session-Based Recommendation". In: *Proceedings of the 1st Workshop on Temporal Reasoning in Recommender Systems co-located with 11th ACM Conference on Recommender Systems*. RecTemp '17. 2017, pp. 50–56

As an extension to [JL17c], this joint work with Iman Kamehkhosh and Dietmar Jannach includes further simple techniques within the previously introduced experimental setup. The author of this thesis contributed to the work by implementing (Association Rules) or proposing (Simple Sequential Rules) these simple techniques, and, furthermore, helped by conducting several of the experiments.

Evaluation of Session-Based Recommendation Algorithms

Malte Ludewig and Dietmar Jannach. "Evaluation of Session-Based Recommendation Algorithms". In: *User Modeling and User-Adapted Interaction* 28.4-5 (2018), pp. 331–390

This joint work with Dietmar Jannach is an extension to [LJJ17] and [KJL17]. The additions consist of proposing a matrix factorization-based model for session-based recommendation and including additional evaluation setups as well as datasets. All extensions were designed, implemented, executed, and evaluated by the author of this thesis. He, furthermore, wrote parts of the text.

Performance Comparison of Neural and Non-Neural Approaches to Session-Based Recommendation

Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. "Performance Comparison of Neural and Non-Neural Approaches to Session-Based Recommendation". In: *Proceedings of the 13th ACM Conference on Recommender Systems*. RecSys '19. 2019, pp. 306–310

This contribution is a joint work with Noemi Mauro, Sara Latifi, and Dietmar Jannach, which is partially based on [LJ18b]. The previous work was extended with the introduction of additional algorithms from the literature. Several of these were implemented by the author of this thesis.

User-Centric Evaluation of Session-Based Recommendations for an Automated Radio Station

Malte Ludewig and Dietmar Jannach. "User-Centric Evaluation of Session-Based Recommendations for an Automated Radio Station". In: *Proceedings of the 13th ACM Conference on Recommender Systems*. RecSys '19. 2019, pp. 306–310

In this joint work with Dietmar Jannach, the most promising techniques from [LJ18b] were evaluated in a user-centric setup. Besides writing parts of the text, the author of this thesis designed, implemented, and conducted the reported user study. In addition, he processed and analyzed the obtained results.

1.6.2 Research Competitions

The author of this thesis, furthermore, successfully competed in several research competitions concerning recommender systems in his time as a research associate. The corresponding publications are presented in the following for the sake of completeness. They are, however, not extensively discussed in this thesis.

A Light-Weight Approach to Recipient Determination When Recommending New Items

Malte Ludewig, Michael Jugovac, and Dietmar Jannach. "A Light-Weight Approach to Recipient Determination When Recommending New Items". In: *Proceedings of the ACM Recommender Systems Challenge 2017*. RecSys Challenge '17. 2017, 3:1–3:6

This publication is a joint work with Michael Jugovac and Dietmar Jannach. The author of this thesis wrote most parts of the text and mainly designed the proposed approach to the 2017 RecSys Challenge on reciprocal job recommendation. In the end, the proposed solution led to the 5th place in an offline evaluation and the 6th place in an online evaluation in the business social network Xing[8].

[8] `https://www.xing.com/`

Could You Play That Song Again? - Reminding Users of Their Favorite Tracks Through Recommendations

Malte Ludewig and Dietmar Jannach. "Could You Play That Song Again? - Reminding Users of Their Favorite Tracks Through Recommendations". In: *Proceedings of the WSDM Cup Workshop 2018*. WSDM Cup '18. 2018

In this joint effort with Dietmar Jannach, besides writing most parts of the text, the author of this thesis mainly designed the proposed approach to the 2018 WSDM Cup for music recommendation. The proposed approach finally led to the 8th place amongst over 1,200 registered participants.

Effective Nearest-Neighbor Music Recommendations

Malte Ludewig, Iman Kamehkhosh, Nick Landia, and Dietmar Jannach. "Effective Nearest-Neighbor Music Recommendations". In: *Proceedings of the ACM Recommender Systems Challenge 2018*. RecSys Challenge '18. 2018, 3:1–3:6

This contribution is joint work with Iman Kamehkhosh, Nick Landia, and Dietmar Jannach. The author of this thesis mainly designed the proposed approach to the 2018 RecSys Challenge for music recommendation, and furthermore wrote most parts of the text. After all, the authors placed 3rd among over 100 competing teams from universities around the world.

Learning to Rank Hotels for Recommendation and Search From Session-based Interaction Logs and Meta Data

Malte Ludewig and Dietmar Jannach. "Learning to Rank Hotels for Search and Recommendation from Session-Based Interaction Logs and Meta Data". In: *Proceedings of the ACM Recommender Systems Challenge 2019*. RecSys Challenge '19. 2019

In this joint effort with Dietmar Jannach, the author of this thesis mainly designed the proposed approach to the 2019 RecSys Challenge on hotel recommendation and wrote most parts of the text. In the end, the authors achieved the 5th place amongst over 500 competitors.

2 Comparison of Session-Based Recommendation Techniques

As mentioned in the introduction, many academic publications in the field of recommender systems focus on techniques that only or mostly rely on long-term preference models. In practical recommendation scenarios, however, such long-term information is often not available for a substantial number of users. They might, for example, be entirely new to a service, or simply access it from another device and are, thus, not logged in. In consequence, an algorithm has to determine suitable personalized recommendations based on the limited amount of information it can access, i.e., the user's few recent interactions with the website or application. Such recommendation techniques are categorized as *session-based* approaches. Amazon's "Customers who bought ... also bought" *item-based* recommendations, where the suggestions are solely based on the currently viewed item, can be seen as an extreme version of such a session-based approach.

As mentioned in Section 1.2, this setting, in academia, is usually operationalized as the problem of predicting the next user action, and the evaluation of novel approaches is performed in offline experiments on large, time-ordered log datasets from various domains. The most prominent domains in this context are the e-commerce and the music domain [QCJ18].

Due to its practical importance, the *session-based* recommendation problem gained increasing interest in the recommender system research community. A large number of novel techniques has been proposed over the recent years. Despite the growing number of published techniques, the papers do not follow any "standard" benchmark in terms of the tested datasets, evaluation protocols, or baseline approaches. Thus, it is often difficult to compare the various proposed algorithms and assess to what extent they contribute to the progress in the field. Furthermore, the algorithmic proposals are predominantly evaluated in terms of common accuracy metrics. Many works, however, showed that there can be huge discrepancies between offline accuracy results and the production performance of a recommender system (see, e.g., [Gar+14]), and it, thus, is advisable to evaluate new techniques also in terms of beyond-accuracy metrics.

To establish a common base for future research, the author of this thesis performed an in-depth multi-dimensional performance comparison across multiple domains and datasets, which involved a number of comparably simple as well as more sophisticated algorithms from the recent literature. At this, a public research framework was developed and shared alongside a diverse selection of datasets from multiple domains. The goal is to lay a foundation for more open and comparable research in the area of session-based recommendation[9]. Furthermore, a number of the compared algorithms were developed and proposed by the author himself throughout the publications [JL17c], [KJL17], [LJ18b], and [Lud+19].

Not only in the area of session-based recommendation but in general the comparison of recommendation techniques is mostly based on *offline* experimentation. As stated in Section 1.3, the number of user studies and field tests to assess the perceived quality of the generated recommendation is comparably low, especially in the rather novel setting of session-based recommendation. Thus, the author of this thesis conducted a between-subjects user study reported in [LJ19b] to address the question of how users perceive the quality of recommendations generated by five different algorithms. The 250 participants were asked to interact with an online radio station application and subsequently answer several questions regarding the recommendation quality in a post-task questionnaire.

This chapter is organized as follows. Next, in Section 2.1 the problem of session-based recommendation, which was only roughly described so far, is formalized. In the following, Section 2.2 provides a detailed technical description of the algorithms that were compared in this thesis. Furthermore, additional related work is presented. Section 2.3 then describes the evaluation setup and Section 2.4 the outcomes of the extensive experiments. Finally, Section 2.5 is dedicated to the comparison of session-based recommendation approaches in the *online* user study. After describing the general setup of the study in Section 2.5.1 and 2.5.2, the main findings are presented in Section 2.5.3.

2.1 Session-Based Recommendation Abstraction

Section 1.2 already informally described and visualized the scenario of session-based recommendation. The main input to the problem is a chronologically ordered list of user interactions recorded in his or her current browsing or application usage session, respectively. Each action refers to a specific type, an item, a user or session identifier, and often a timestamp. When visiting an online shop, the user, e.g., clicks on a certain t-shirt, favors it on a wish-list, or adds it to the cart. On a music platform,

[9] The framework can be found on GitHub: `https://github.com/rn5l/session-rec`

for example, he listens to a song or skips it. Furthermore, a few datasets also contain metadata for the items, users, or the context of the interactions. The output is simply defined as an ordered recommendation list.

To test approaches uniformly across multiple domains, the problem formulation has to be standardized. Independent of the domain, all scenarios have in common that a user always shows interest in an item with at least one interaction type, e.g., he plays a track, he starts a video, he clicks on an item in an e-commerce shop, or views a hotel detail page in a mobile booking application. To overcome the problem of multiple and different types of interactions, the main input is here reduced to only the most common action type, which shows the users' interest in an item. Furthermore, as timestamps and metadata are not constantly available across all datasets, they are also neglected.

This slightly abstracted problem of session-based recommendation can be defined as follows: Let a session s be a chronologically ordered list of m items $s = [s_1, s_2, s_3, \ldots, s_m]$, S_p the set of all past sessions, and I the unique set of all items. Given a user's current session s, a session-based approach defines a function

$$score_{session}(s, i, S_p) \tag{2.1}$$

which returns a relevance score for each item i in I with regard to the current session s. A recommendation list of variable size can now be retrieved by sorting the items according to the $score$.

2.2 Technical Approaches

Algorithmic approaches to session-based recommendation are usually designed to predict the next item given the session beginning. Early proposals were, e.g., based on association rules, sequential pattern mining, or nearest neighbors [Mob+02; YLY12; HMB12]. Subsequently, more complex approaches of different types were applied to the problem, e.g., factorized Markov models [RFS10]. Recently, deep learning approaches mainly dominate the literature [Hid+16a; Li+17; Liu+18].

In the following, this thesis will first provide an in-depth description of all approaches that are included in the unified research framework. These can be classified into the categories frequent-pattern mining, nearest-neighbor techniques, factorization models, and neural networks. In the end, further related methods will be discussed to provide a comprehensive overview.

2.2.1 Frequent-Pattern Mining

As simple baseline techniques, the author of this thesis implemented three straightforward frequent-pattern mining approaches: an association rule technique, first-order Markov Chains, and a heuristic method based on sequential rules. All approaches have in common that their computational complexity is very low, both in the training and the recommendation phase. Furthermore, the framework includes a recent more complex algorithm that adapts context trees from the field of file compression to represent frequent sequences in the data [MF18].

Association Rules

Simple Association Rules (AR) implements a simplified version of association rule mining with a maximum rule length of two. The method is designed to recommend items in the style of "Customers who bought ... also bought".

Technically, the association rules are extracted by simply counting the co-occurrences of two items i_1 and i_2 in all sessions. Thus, given a user's current session s with $s_{|s|}$ being the last item in s, we can define the session-based score function as:

$$score_{\text{AR}}(s, i, S_p) = \frac{1}{\sum_{p \in S_p} \sum_{x=1}^{|p|} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot (|p| - 1)} \sum_{p \in S_p} \sum_{x=1}^{|p|} \sum_{y=1}^{|p|} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot 1_{\text{EQ}}(i, p_y) \tag{2.2}$$

The indicator function $1_{\text{EQ}}(a, b)$ is 1 in case a and b refer to the same item and 0 otherwise. In Equation 2.2, the co-occurrences are first counted (right side) and then normalized by the number of rule occurrences for the current item $s_{|s|}$.

Markov Chains

This simple approach (MC) can be seen as a sequential version of AR. The rules capture the transition probabilities between two subsequent items in a session and, thus, represent a first-order Markov Chain. The baseline is implemented by simply counting how often users viewed an item i_2 immediately after another item i_1. Consequently, the score can be calculated as:

$$score_{\text{MC}}(s, i, S_p) = \frac{1}{\sum_{p \in S_p} \sum_{x=1}^{|p|-1} 1_{\text{EQ}}(s_{|s|}, p_x)} \sum_{p \in S_p} \sum_{x=1}^{|p|-1} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot 1_{\text{EQ}}(i, p_{x+1}) \tag{2.3}$$

Sequential Rules

The third baseline method SR is a variation of MC or AR respectively and was proposed by the author in [KJL17]. The order of the items in the session is also taken into account, but in a less restrictive manner. A rule is created when an item i_2 occurs at some point after an item i_1 in a session. The weight of a rule is determined by the distance in terms of the number of events between the two items, e.g., with: $w_{\text{SR}}(d) = 1/(d)$, where d is the number of interactions between the two items.[10] Following the previews formulas the score function for SR is defined as:

$$score_{\text{SR}}(s, i, S_p) = \frac{1}{\sum_{p \in S_p} \sum_{x=2}^{|p|} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot x} \sum_{p \in S_p} \sum_{x=2}^{|p|} \sum_{y=1}^{x-1} 1_{\text{EQ}}(s_{|s|}, p_y) \cdot 1_{\text{EQ}}(i, p_x) \cdot w_{\text{SR}}(x-y) \tag{2.4}$$

Context Trees

In contrast to MC, which can be seen as a simple first-order Markov chain, the context-tree-based approach proposed in [MF18] represents a variable-order Markov model. The technique aims to capture sequential patterns in sessions that occur over multiple interactions. Furthermore, it is non-parametric, i.e., it is not dependent on any parameters that require an optimization.

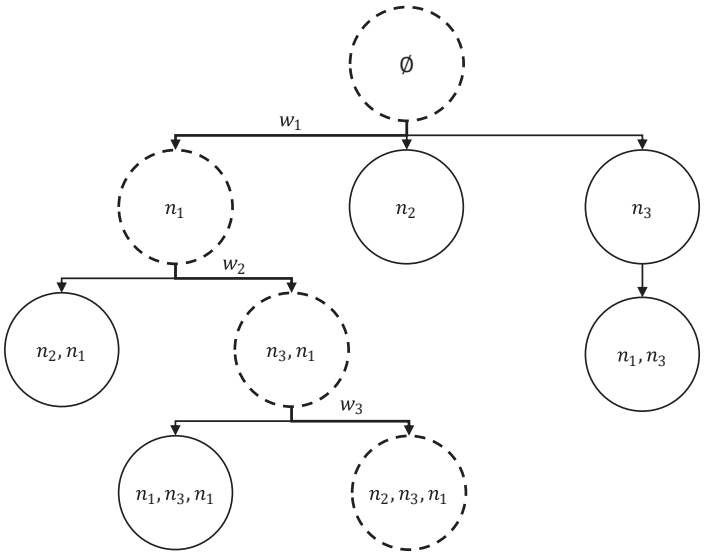


Figure 2.1: Visualization of a simple context tree. Given a current user session (n_2, n_3, n_1), the dashed nodes would be traversed top-down in the process of creating recommendations (adapted from [MF18]).

[10] Other weighting functions, e.g., a logarithmic decay, are possible as well. Using the linear function, however, on average led to the best results in the experiments.

In the training phase, a context tree is constructed from all users' historic sessions. Figure 2.1 presents a minimal example of such a tree, which was constructed from just a few example sessions. Given a current user session (n_2, n_3, n_1), the tree is now traversed top-down and nodes in the tree are "activated" (dashed nodes) when the node represents a suffix of the current session. While building the tree, each node records the probability that a certain item occurs after the sequence that the node is representing. These probabilities are then combined in a weighted scheme for all nodes activated by the current session to rank potential items from the catalog. Note that the weights (w_1, w_2, w_3) in Figure 2.1 are automatically determined in the training or the tree construction phase, respectively.

2.2.2 Nearest-Neighbor Techniques

In previous works (see, e.g., [VG14]), context-aware nearest-neighbor-based approaches often showed a surprisingly good performance in terms of their predictive accuracy. Hence, the author of this thesis adapted this scheme and refined it in multiple nearest-neighbor techniques, which were proposed throughout [JL17c], [LJ18b], and [Lud+19]. Furthermore, also traditional item-based schemes are suitable for the session-based problem setting and have been employed as a baseline method in previous works [Hid+16a].

Item-Based Nearest Neighbors

Similarly to AR, the item-based kNN method (IKNN) implements the "Customers who bought ... also bought" scheme. Similarly, IKNN solely bases its recommendations on the last given item in the current session. Then, the recommendations are formed from the items that are most similar to the last given item in terms of the co-occurrences in other sessions. The items are encoded as binary vectors over the session-space, i.e., each position of the vector corresponds to a certain session and is set to 1 when the item is part of that session. Item similarities are then calculated by measuring the cosine similarity between the vectors. In contrast to AR, the application of the cosine similarity measure makes the approach less prone to popularity biases. In general, despite their simplicity, item-to-item approaches are often used in practice and usually show a strong performance [LSY03; Dav+10].

Session-Based Nearest Neighbors

The basic session-based kNN technique (SKNN) (see [BJ14]) shares some similarities with IKNN. However, in contrast to solely basing the recommendation on the very last item in the current user session, SKNN considers all items in the session. Then, instead of directly finding similar items, SKNN determines the k most similar past sessions. At this, sessions are represented as binary vectors over the item-space. A set of neighboring session N_s is then calculated with a suitable similarity measure, e.g., again, the cosine similarity. Given the current session s, the neighbors N_s, and a similarity measure $sim(s_1, s_2)$, the score function can be defined as:

$$score_{\text{SKNN}}(s, i, S_p) = \Sigma_{n \in N_s} sim(s, n) \cdot 1_n(i) \tag{2.5}$$

In this case, the function $1_n(i)$ indicates if the session n contains the item i.

Scalability. Although nearest-neighbor-based methods usually show a solid performance, they are very prone to scalability issues. In a large scale scenario, it is computationally not feasible to calculate the similarities of the current session to all past user sessions. In [JL17c], the author of this thesis, thus, presented an implementation of the algorithm that is applicable in practical scenarios. The improvements rely on fast in-memory data structures and heuristic neighborhood sampling (refer to [JL17c] for more technical details). It has shown that sampling a small fraction of all past sessions only leads to small decreases in the prediction accuracy. Capturing recent trends in the community by sampling solely the most recent sessions often even proved to be beneficial. The nearest-neighbor implementations, therefore, have an additional parameter m, which determines the size of the sample. In the experiments reported in the course of this thesis, it was, for example, mostly sufficient to consider only the 1,000 most recent sessions that include one item from the current session at minimum.

Sequence-Awareness. The basic SKNN approach does not incorporate the order in which the items occur in a session. However, the order of the elements might be of particular importance, e.g., as the focus or the goal of the user might shift over a single session. Thus, in [LJ18b], the author of this thesis proposed several sequence-aware extensions to SKNN.

- **Vector Multiplication Session-Based kNN (V-SKNN)**: The main idea of this extension is to consider more recent events in a session as more important. Instead of representing the current user session as a binary vector, a real-valued weighted encoding is performed. While only the very last interaction is assigned the value 1, the previous interactions obtain a lower weight determined by a variable decay function, e.g., a linear, logarithmic or exponential decay based on

the distance to the most recent interaction in steps. The session similarity is then calculated as the dot product. Note that past sessions are still encoded as binary vectors over the item-space.

- **Sequential Session-Based kNN (S-SKNN)**: The second extension, again, emphasizes items more the later they occur in the current user session. In this case, however, the final scoring function is adjusted:

$$score_{\text{S-SKNN}}(s, i, S_p) = \Sigma_{n \in N_s} sim(s, n) \cdot w_{seq}(i, n, s) \tag{2.6}$$

  To reward items that appear later in the current session, the indicator function $1_n(i)$ is replaced with a weighting function $w_{seq}(i, n, s)$. For a neighbor session n, the weight is higher the more recently an item was consumed in the current session that appears in both of the sessions. Following V-SKNN, if the most recent item $s_{|s|}$ appears in both sessions, the assigned weight is 1. Otherwise, the weight is again determined by a variable decay function based on the distance d to the most recent item in terms of the steps, e.g., $w_{seq}(i, n, s) = (|s| - d)/|s|$.

- **Sequential Filter Session-Based kNN (SF-SKNN)**: The third extension, like S-SKNN, also adjusts the scoring function, but in a more restrictive way. Here, the recommendation of an item is only allowed in case it appears after the most recently consumed item at least once in the dataset.

$$score_{\text{SF-SKNN}}(s, i, S_p) = \Sigma_{n \in N_s} sim(s, n) \cdot 1_n(s_{|s|}, i) \tag{2.7}$$

  The scoring function is almost identical to SKNN, however, the indicator function $1_n(s_{|s|}, i)$ is more restrictive. As stated above, it only returns 1 if the item i occurs immediately after $s_{|s|}$ in any session s in S_p.

In the course of the author's most recent work [Lud+19], all sequential extensions have been integrated into a single parametrized approach, which is hereafter referred to as V-SKNN. Furthermore, [Lud+19] presented an additional extension that favors the recommendation of less popular items to mitigate potential popularity biases. At this, the inverse document frequency (IDF) is calculated for each item and included in the final scoring function:

$$score_{\text{V-SKNN}}(s, i, S_p) = \Sigma_{n \in N_s} sim(s, n) \cdot w_{seq}(i, n, s) \cdot w_{idf}(i) \tag{2.8}$$

Here, $w_{idf}(i)$ adds an IDF-based weight of configurable strength, which showed a beneficial effect for the successful generation of playlist continuations in previous experiments conducted by the author of this thesis [Lud+18].

2.2.3 Factorization-Based Approaches

As mentioned in the beginning of this section, the second category of techniques that can be applied in the session-based scenario are a number of factorization-based methods. Besides sequence-aware factorization approaches, also the basic implicit feedback recommendation approach Bayesian Personalized Ranking (BPR) has been included as a baseline and adapted to the scenario. Furthermore, three sequential methods from the literature have been included, Factorized Personalized Markov Chains (FPMC) [RFS10], Factored Item Similarity Models [KNK13], and Factorized Sequential Prediction with Item Similarity Models [He+16]. Like BPR, those techniques are not explicitly designed for the scenario of session-based recommendation and were slightly adjusted. Finally, the author of this thesis designed a novel technique named Session-based Matrix Factorization, which adopts ideas from previous techniques but is, however, created specifically for the given scenario.

Bayesian Personalized Ranking

Bayesian Personalized Ranking (BPR) is well-known and widely used as a baseline approach in the recommender systems research community [Ren+09]. Although the method is not designed for the scenario of session-based recommendation, it can easily be adapted to the setting and has, thus, been included as a baseline technique, representative for traditional long-term preference models. To apply BPR in the given setting, each session was assumed to be caused by a unique user, i.e., the session corresponds to the user in a sparse user-item interaction matrix. After applying BPR, each user and item is represented by a lower-dimensional vector of latent factors. Given the current session s, recommendations can be generated by inferring a user vector u_s as the average of all latent vectors for the items in s. Traditionally, the score for item i is determined by calculating the dot product of the user profile u_s and the latent factors for i.

Factorized Personalized Markov Chains

Originally, Factorized Personalized Markov Chains (FPMC) were designed for the problem setting of next-basket recommendation [RFS10]. Given a history of purchased shopping carts, the task is to predict the item that a user will purchase next. By simply limiting the maximum basket size to one and assuming the current session to be the history of baskets, the approach immediately suits the session-based recommendation scenario.

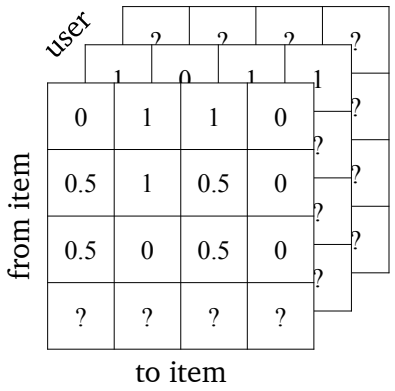


Figure 2.2: Visualization of the personalized transition cube (adapted from [RFS10]).

From an algorithmic perspective, FPMC extends the traditional user-item matrix factorization by a third dimension, the item-item transition probability (see Figure 2.2). To factorize the three-dimensional tensor, internally a specific form of Canonical Tensor Decomposition is applied. Again, as no long-term user histories are available in this scenario, the user latent vectors are estimated as the average of the item latent vectors in the current session. More technical details can be found in [RFS10] and [LJ18b] respectively.

Factored Item Similarity Models

In contrast to FPMC, Factored Item Similarity Models (FISM) rely solely on item-item factorization and, thus, are directly applicable in the session-based scenario. The score of an item i is determined by the sum of latent vector products between i and each item that was already "rated" by a user. Originally, in [KNK13], the model was designed to predict ratings, for example, in the movie domain. Thus, for the given session-based scenario, the training procedure from BPR was adapted to find an optimal ranking for all items given the items in a session. Even though the model is directly applicable in the given setting, a possible disadvantage is that sequential item-item relations in the data are neglected. Technical details can be found in [KNK13] and [LJ18b], respectively.

Factorized Sequential Prediction with Item Similarity Models

The Factorized Sequential Prediction with Item Similarity Model (FOSSIL) tries to overcome the previously mentioned disadvantage of FISM by incorporating sequential information in the form of factorized Markov chains [He+16]. The FISM model is

extended by a second item-item factorization part, which is supposed to learn an item-item transition probability. To determine this second part of the score for a certain item i, a latent representation of i is multiplied with a latent representation of the last item in the current session. Finally, the second score is combined with the FISM score in a weighted sum (see [He+16] and [LJ18b] for details). Again, the model is directly applicable in the session-based scenario as no long-term information is needed for prediction. The training of the model is performed similarly to FISM.

Session-Based Matrix Factorization

Finally, with the Session-based Matrix Factorization technique (SMF), the author of this thesis proposed a novel factorization-based model that was specifically designed for the task of session-based recommendation. The basic idea is to combine traditional matrix factorization with factorized Markov chains while considering the special cold-start scenario of session-based recommendation.

In a simplified traditional factorization-based prediction model, the scores for a user u and an item i are determined with

$$score_{factorized}(u, i) = p_u q_i^T \tag{2.9}$$

p_u and q_i represent the lower-dimensional latent vectors for user u and item i, respectively. As the user is anonymous, in SMF the vector p_u is replaced by a session preference vector s_e, which is calculated in the following way:

$$s_e = M_{ST} \cdot s_b^T \tag{2.10}$$

Here, s_b is a binary representation of the current session s (see Section 2.2.2), and M_{ST} is a transformation matrix that maps s to the latent vector space. Thus, M_{ST} is of size $|I| \times |q_i|$.

Using the transformed session representation s_e, the score function of SMF is presented in Equation 2.11.

$$score_{\text{SMF}}(s, i, S_p) = \underbrace{s_e q_i^T + b_{1,i}}_{\text{session preferences}} + w_i \cdot (\overbrace{n_{s_{|s|}} m_i^T + b_{2,i}}^{\text{sequential dynamics}}) \tag{2.11}$$

Here, the $score_{\text{SMF}}(s, i, S_p)$ for a session s with the most recent item $s_{|s|}$ is calculated as a weighted sum of the session preferences and sequential dynamics. The session preferences s_e simply replace the long-term user vector in the traditional matrix

completion setup and the score for an item i with respect to the current session s is defined by the dot product of s_e and the latent item vector q_i. The sequential dynamics are determined like in FOSSIL, i.e., a latent vector representation of the item to score m_i is multiplied with a representation of the last item in the current user session $n_{s_{|s|}}$. Furthermore, both parts of the overall model include a bias term $b_{x,i}$ to learn a global unpersonalized score for all items in the dataset. The weighting w_i, finally, is also dependent on item i.

The model can be trained by applying a stochastic gradient descent (SGD) optimization procedure and similarly to BPR learns to rank positive examples of session continuations (subsequent items to the given session start) over negative ones (random items that do not occur in the session). Furthermore, regarding the exact training procedure and the loss functions, some improvements from [HK18] were included in the approach (see [LJ18b] for details).

2.2.4 Neural Networks

Due to the increasing success of deep learning in various fields and the growing relevance of the session-based recommendation scenario, over the recent years numerous session-based techniques based on neural networks were proposed. Early approaches relied on Recurrent Neural Networks (RNNs), which showed great success in the area of neural language processing (NLP) and seem like a natural choice to model the sequence of actions in a user session [Hid+16a; HK18]. Later, also alternative network architectures were adopted for the session-based recommendation task as they were reported to be superior to RNNs for certain task, e.g., convolutional neural networks [Yua+19], attention networks [Liu+18], memory networks [Wan+19], or graph-based networks [Wu+19].

In the course of the work described in [JL17c], [LJ18b], and [Lud+19], the author of this thesis implemented various approaches in the unified framework, which was publicly released alongside [Lud+19]. To represent the most common network architectures, these include an early approach based on Gated Recurrent Units (GRUs), a convolutional neural network, and a recent approach that relies solely on the attention-mechanism concept. For the sake of completeness, also more recent works that could not be included in the comparison due to time restrictions are discussed in Section 2.2.5.

Gated Recurrent Units for Recommendation

Techniques based on RNNs represent the first deep learning approaches that were applied to the scenario of session-based recommendation. The most prominent one uses Gated Recurrent Units for Recommendation (GRU4REC), was first presented in [Hid+16a], later on improved in [HK18], and is designed to predict to probability fo

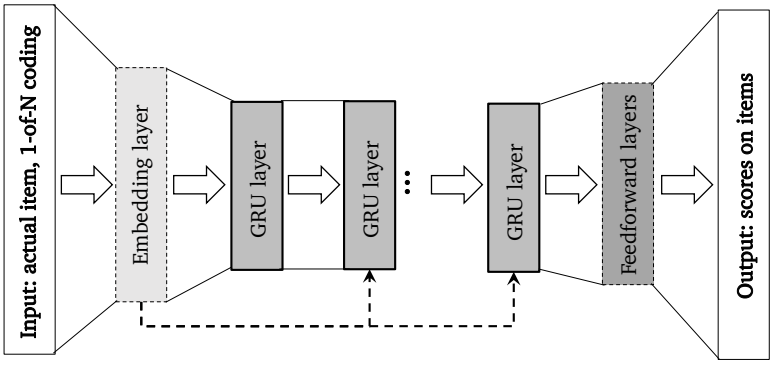


Figure 2.3: Architecture of the GRU4REC neural network (adapted from [Hid+16a]).

The overall architecture of GRU4REC is visualized in Figure 2.3. The input to the network is a single item, which is one-hot encoded in the item-space and the output is a similar-shaped vector that represents a probabilistic distribution for the next item given a session beginning. The embedding layer, the feedforward layer, and additional GRU layers, hereby, are optional.[11] The in-between GRU layers memorize a lower-dimensional hidden state that represents all items that a user has previously interacted with in a session.

The choice of applying RNNs for session-based recommendation might be natural, however, with GRU4REC the authors introduced further innovations for the specific setting of session-based recommendation. In [Hid+16a] they proposed an efficient training procedure, and in [HK18] they furthermore presented new loss functions that helped to increase the predictive performance significantly.

Neural Attentive Recommendation Machine

The Neural Attentive Recommendation Machine (NARM), as proposed in [Li+17], follows the general idea of GRU4REC to employ RNNs, GRUs in particular, to model the user sessions. However, as an additional attention-mechanism showed to improve

[11] Indeed, in [Hid+16a] the authors report that a single GRU layer showed the best performance.

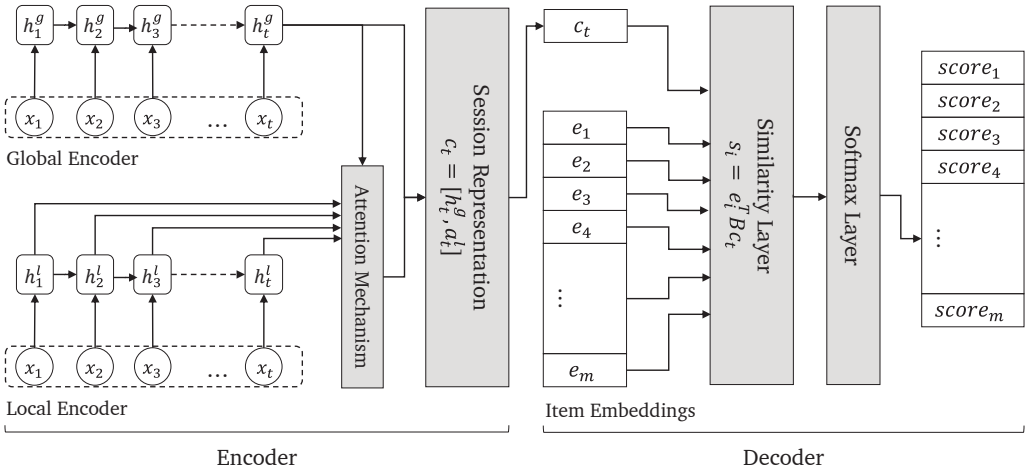


Figure 2.4: Architecture of the NARM neural network (adapted from [Li+17]).

the performance of recurrent networks in NLP [BCB15], the authors of NARM transferred this principle to an attentive network architecture for the scenario of session-based recommendation.

Figure 2.4 gives an overview of the NARM architecture and shows how the attention-mechanism is applied. The *Global encoder* directly corresponds to the concept of GRU4REC, the *Local encoder*, however, adds the attention-mechanism from [BCB15]. Instead of just encoding the session by the last hidden state of the RNN (h_t^g), all hidden states are considered. This is achieved by calculating a similarity between the last "global" hidden state h_t^g and all previous "global" hidden states h_n^l. Then, the final *local* encoding of the session a_t^l is simply determined as the sum of all hidden states, weighted by their similarity to h_t^g. In the process of determining the item scores, NARM, in comparison to GRU4REC, furthermore adds a *Similarity Layer*, which is finally softmax-activated. Following traditional matrix factorization techniques, the *Similarity Layer*, similar to SMF, calculates a dot product between the final session representation c_t (concatenation of a_t^l and h_t^g) and the latent item representations e_i. For more technical details, refer to [Li+17].

Short-Term Attention/Memory Priority Model

A disadvantage of RNNs is that the calculation of the hidden states has to be performed step-by-step, i.e., the next state depends on the previous one. Thus, no parallelization can be applied and the computations can become demanding for longer sequences. The Short-Term Attention/Memory Priority Model (STAMP) tries to overcome this problem by completely neglecting RNNs and solely relying on the previously introduced attention-mechanism [Liu+18].

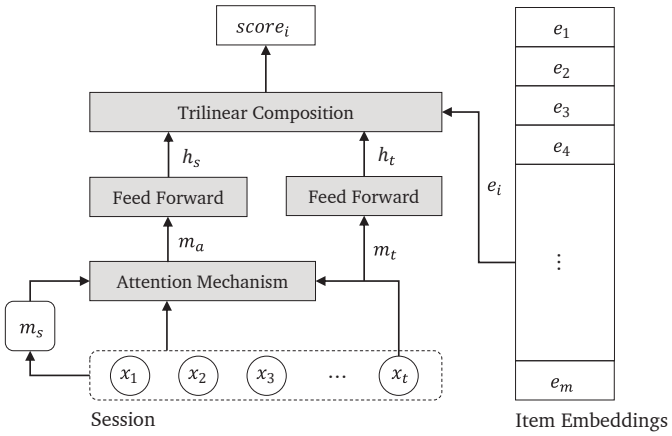


Figure 2.5: Architecture of the STAMP neural network (adapted from [Liu+18]).

Figure 2.5 presents an overview of the STAMP architecture. Each item x_i in the current session is represented by a lower-dimensional latent representation (embedding layer). Instead of first applying an RNN, as in NARM, the attention-mechanism is directly applied to the embedded items to create a session representation m_a. Here, not the last hidden state but simply the last embedded item x_t is used as a reference point for the calculation of the similarity to each item in the session. Differently from the default attention mechanism described for NARM, the similarity function in STAMP furthermore includes a mean vector of all items in the session (m_s). The session representation m_a and the last item in the session x_t are then fed into individual feed-forward layers to create two state vectors h_s and h_t. Finally, the score for an item i is determined in a triple product of these two states and a latent (embedded) representation e_i for the target item i.

Convolutional Generative Network for Next-Item Recommendation

Similar to STAMP, which applies a pure attention-based architecture, the Convolutional Generative Network for Next-Item Recommendation (NEXTITNET) relies on the concept of convolutional networks to overcome the performance disadvantages of RNNs [Yua+19]. Convolutional neural networks (CNNs) are the preferred architecture for problems related to computer vision. They, however, also showed to be successful for the task of modeling a sequence of signals, e.g., in [Oor+16].

Figure 2.6 visualizes the general concept of applying CNNs to the task of session modeling. First, as in STAMP, the items in the session (x_1 to x_{11}) are represented by lower-dimensional embedding vectors. Here, the resulting matrix can be seen as an "image" of the current session. Next, traditional convolutional layers are applied to "scan" the session. A pooling layer transforms the scanned information

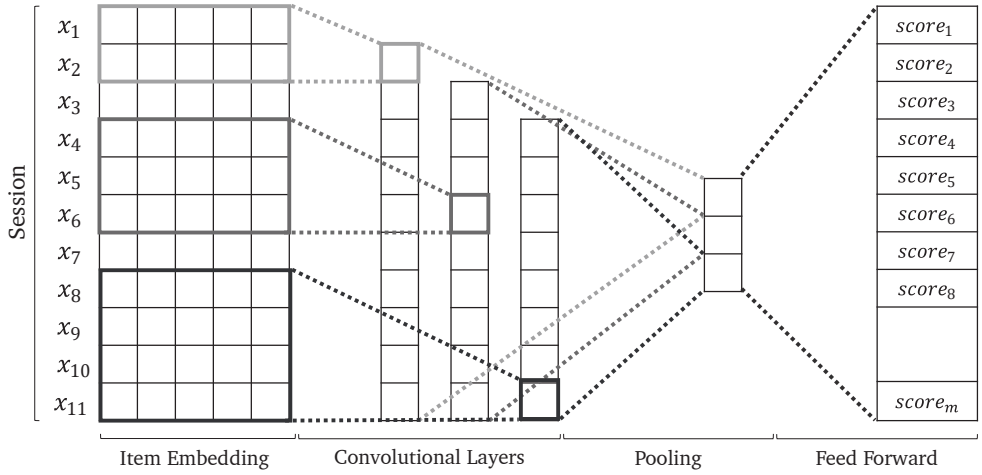


Figure 2.6: Abstract architecture of convolutional neural networks for session-based recommendation (adapted from [Yua+19]).

into a one-dimensional vector, from which, with the help of a feed forward layer, a probability distribution for the next item can be predicted. In general, NEXTITNET utilizes this architecture but, furthermore, introduced several improvements, e.g., from [Oor+16]. Mainly, the authors transform the session "image" into a one-dimensional representation to be able to apply more effective 1-D convolutions. Furthermore, they incorporate the concept of dilated convolutions with "holes" to improve the modeling of longer sequences [Oor+16], and, finally, borrow concepts of residual learning to increase the model stability [He+15].

2.2.5 Further Related Work

Besides the representative methods that were included in the experimental setup, for most categories of session-based techniques, e.g., neural models, a multitude of other related approaches has been proposed in the literature over the recent years. This section provides an overview for the sake of completeness.

Historically, early approaches that are applicable to session-based recommendation mostly rely on some form of frequent pattern mining, e.g., in [Mob+02] the authors try to predict the next step in user navigation behavior. Also in the music and e-commerce domain similar techniques were then applied for predicting the next item in various recommendation scenarios [YLY12; HMB12; BJ14].

Besides GRU4REC, NARM, STAMP, and NEXTITNET the majority of the recent works that specifically address the session-based recommendation scenario are neural-network-based techniques, e.g., [Wu+19] or [Wan+19]. After the authors of GRU4REC first introduced their RNN-based approach, a few improvements to the

model have subsequently been proposed in [TXL16]. However, with the extensions presented by the original authors in [HK18], the latest version of GRU4REC showed a superior predictive performance again. In [LLH17], the authors, similar to NARM, presented an alternative architecture based on RNNs with the attention-mechanism that was first introduced for machine translation in [BCB15]. In [GWD14] and [WCB15], external memory modules were proposed for neural networks. The authors of [Wan+19] incorporated this concept for session-based recommendation by introducing a memory of the most recent historical user sessions to an RNN. Following a concept similar to SKNN, the prediction of the next item then additionally relies on the most similar past session. With the ideas presented in [Sca+09], neural networks became also popular for graph-based tasks. In [Wu+19], the authors model user sessions as graphs and rely on such graph neural networks alongside an attention-mechanism to model the current session and predict the next item. A particularity of session-based or, in general, sequence-aware recommendation is that it can be beneficial to repetitively recommend items that a user is already aware of (see Section 1.2). RepeatNet [Ren+19] addresses this aspect of session-based recommendation. The authors, therefore, propose an RNN-based approach that explicitly tries to model if a user is in an exploratory mood and, thus, should be presented with unknown items, or it is likely that he will click on an item again. A number of neural session-based approaches furthermore showed that the integration of timestamps and content data, or the consideration of multiple interaction types in the network architecture can be beneficial. In [BK17], for example, the authors considered item dwell times in a GRU4REC-like RNN model. The authors of GRU4REC, in [Hid+16b], presented an extended architecture, which showed to improve the predictive accuracy by incorporating video thumbnails or product images that were encoded with CNNs. In the news domain, e.g., word embeddings were employed alongside CNNs to create article representations [SFC18]. These, in turn, were fed into an Long Short-term Memory (LSTM) network designed for the session-based recommendation of suitable articles.

Furthermore, the SKNN technique presented by the author in [JL17c] also led to a number of follow-up publications, e.g., [Guo+19] and [Gar+19]. In [Guo+19], the authors proposed two extensions to SKNN, which add sequential constraints to the session sampling procedure and account for the item popularity in the session similarity calculation. Similar to V-SKNN, [Gar+19] proposes several sequential extensions to the basic SKNN method. As in V-SKNN, the authors give more weight to items that have been more recently clicked in the current user session. Furthermore, their approach utilizes timestamps to weight neighboring sessions by recency and considers the position of candidate items in neighboring sessions. The items obtain a higher relevance score, the closer they have been clicks to an item from the current session. Finally, in [SSZ18] the authors explored the utilization of content data in techniques similar to SKNN in the news domain.

Besides the mentioned session-based approaches, a number of related works exist that focuses on predicting the next item and incorporating sequence information but not explicitly in a session-based scenario. Such models are often based on Markov chains [He+09; ML11; GDF13; Hos+15], Markov decision Processes (MDP) and Reinforcement Learning (RL) [SHB05; MBR12; TB14], or neural networks [Zha+14; Sor+15; SEH16; Twa16; Yu+16; Du+16; Soh+17]. The typical application scenarios, again, are the music and e-commerce domain. However, most of the approaches require long-term histories or additional item metadata, and are, thus, not included in the in-depth comparison.

Dense item representations are another category of sequence modeling techniques. They represent items in lower-dimensional vectors that are meant to encode sequential information in the data, e.g., with latent Markov embeddings [Che+12; CXJ13; Wu+13; Fen+15] or distributional embeddings [Dju+14; Bae+15; Grb+15; Tag+15; VSC16; RLJ16; Zhe+10]. A comprehensive discussion of approaches for the more general problem setting of sequence-aware recommendation can be found in [QCJ18]. In the context of [LJ18b], however, the author of this thesis also tested various item embedding techniques (see [Ren+09; Mik+13; PSM14]) as an alternative way of representing sessions. As, e.g., in [Tag+15], the lower-dimensional vectors can be used as an alternative to the binary representation in SKNN and IKNN. Using such embeddings, however, did not lead to improvements over the original nearest-neighbor techniques.

2.3 Evaluation Scheme

Section 1.3 introduced the most common evaluation schemes in the areas of session-based and session-aware recommendation. In this section, the protocol that was employed in the in-depth comparison conducted by the author of this thesis is presented in more detail (see also [LJ18b]).

2.3.1 General Setup

As stated in Section 1.3, the main output of the session-based recommendation problem is a ranked list of items that matches the given session beginning. The "gold standard" for evaluation in the research literature is usually to withhold items in test sessions that an algorithm then has to predict.

Most recent proposals in the area of session-based recommendation rely on an incremental *given-N next-item* scheme (see Section 1.3). At this, iteratively all items in a session are chronologically revealed and the corresponding immediate next item

forms the ground truth for prediction. This scheme was chosen for the in-depth comparison, as it a) includes alternative protocols with a fixed N and b) represents the users' journey throughout a session well.

From a logical point of view, however, the immediate next item might not be the only relevant item to the user. Also, other held-out items of the same session are apparently of interest to the user and, thus, should eventually be included in the ground truth.The comparison presented in this thesis, hence, additionally includes measurements using an incremental *given-N remaining items* scheme.

Accuracy Measures. Dependent on the evaluation protocol, the following accuracy measures were included:

- In the *given-N next-item* setup, the ground truth consists of the *immediate next item*. The metrics that are commonly used in the literature (see [Hid+16a; Liu+18]) and, thus, were included are a) the hit rate (HR) and b) the Mean Reciprocal Rank (MRR) at defined list lengths. While the ranking of the recommendations is not important for the HR, the MRR rewards the position in which a relevant item is recommended. The results are averaged over all test sessions.
- In the *given-N remaining items* setup, as mentioned in Section 1.3, all standard information retrieval measures can be applied. Here, the comparison includes Precision as well as Recall as non-ranking measures and the Mean Average Precision (MAP) as a ranking-aware metric.

Splitting Procedure. As explained in Section 1.3, the majority of the publications relies on a single *session-level timed-based split* to separate the full dataset into a training and test part, e.g., [Hid+16a], [Li+17], and [Liu+18]. In order to ensure comparability with other works, the exact same setup was also included in the conducted experiments.

However, single-split protocols have an apparent limitation. The results that are collected for one specific train-test configuration could be prone to random effects or particularities of the data. Thus, it is generally advisable to apply some sort of *cross-validation* scheme. As the data is chronologically ordered, a *sliding-window* protocol was used. The data was split into 5 slices of equal size, which, then again, was timed-based split into training and test parts. A more detailed description is presented in [LJ18b].

Additional Quality Factors. In offline experiments regarding session-based recommendation, researchers often neglect the underlying purpose of a recommendation technique [JA16], e.g., if the system should be designed for the discovery of new

and maybe unpopular items or for finding substitutes to an inspected item. As a consequence, accuracy should not be the only quality factor that is tested for algorithms in an offline evaluation. To account for such scenarios, the following additional quality measures were included:

- *Coverage:* First, it was measured how many different items overall appear in the top-k recommendation lists produced by an algorithm. The metric is sometimes also named *aggregate diversity* (see [AK12]) and should ensure that an algorithm not only focuses on a few items of the catalog.

- *Popularity:* Dependent on the data, obtaining a high accuracy can correlate with the recommendation of mostly very popular items (high popularity bias). The popularity tendency of an algorithm should, thus, be measured as well, because such recommendations might not be satisfactory to the users. The popularity score for an algorithm is calculated as the mean global popularity (number of item occurrences divided by the total number of events) of the items in all generated recommendation lists.

- *Scalability:* Modern deep learning models are often complex and require a substantial amount of computational power for the training process. Furthermore, the extensive tuning of hyper-parameters is often required to even achieve satisfactory results with the models. Thus, all techniques were also tested in terms of their memory requirements and runtimes, both for the training as well as the recommendation phase.

Overall, these beyond accuracy metrics should contribute to a better understanding of certain characteristics of the tested recommendation techniques and reveal possible practically relevant particularities and limitations.

Parameter Optimization. Some of the tested approaches, especially the neural networks, have many hyper-parameters that can affect the training process and, more importantly, the accuracy performance significantly. Thus, for each dataset the hyper-parameters for these techniques must optimized in a consistent procedure to ensure reliable results and, most of all, a fair comparison. Following the procedure in [Hid+16a], the hyper-parameters were determined through a random search of the parameter-space in 100 iterations. Performing an exhaustive grid search with a set of reasonable parameter combinations, for many of the techniques, is impossible in a reasonable amount of time due to the computational complexity of the methods (see also Section 2.4.2). As in [Hid+16a], the hyper-parameters were tuned with respect to the MRR at a list length of twenty. The exact optimization procedure, the considered parameter-space per technique, and the final parameters can be found in [LJ18b] as well as [Lud+19] and the corresponding appendices.

2.3.2 Explored Datasets

The experiments performed by the author of this thesis throughout [JL17c], [LJ18b], and [Lud+19] include nine datasets from three different domains. These originate mostly from the e-commerce websites and music services. Furthermore, data from a large news website was included to also represent a fast-moving domain, in which items are only relevant for a very short time. Table 2.1 provides a short description of all datasets. A more detailed discussion, also regarding the particularities of the domains, is presented in the included publications.

Table 2.1: Brief description of the datasets included in the comparison.

	E-Commerce
RSC15	This is one of the standard datasets in session-based recommendation and was also used in most of the related works [Hid+16a; Li+17; Liu+18]. The data was released in the course of the 2015 ACM RecSys Challenge. It contains timestamped click sequences of sessions recorded over 6 months. Besides item views, the data also includes purchases.
RETAILR	In the context of a Kaggle competition, the personalization company *retailrocket* provided a dataset that, again, consists of user logs collected over six months.
DIGINETICA	This dataset was published in the context of the 2016 CIKM Cup by the e-commerce personalization company *Diginetica*. Like RSC15 and RETAILR, the data contains multiple different action types and was reduced to just item inspection events.
ZALANDO	In contrast to the other datasets, the last e-commerce dataset is non-public. The data consists of user logs of the online fashion retailer Zalando collected over one year.
	Music
8TRACKS	This non-public dataset was provided by the online playlisting service 8tracks. The data includes handcrafted playlists.
AOTM	This dataset also consists of handcrafted playlists that were collected from the Art-of-the-Mix platform. In contrast to *8TRACKS*, *AOTM* is publicly available [ML12].
30MUSIC	This dataset consists of historic user listening logs of the online music service last.fm and was published in [Tur+15].
NOWPLAYING	The last music dataset was constructed from tweets with the hashtag "#nowplaying", in which users post the tracks they are currently listening to. Thus, *NOWPLAYING* represents another dataset of music listening logs [Zan+14].
	News
CLEF	This dataset was released by the advertising company *plista* for the 2017 CLEF NewsREEL challenge.[12] The data includes article-read logs from multiple news platforms. The most interactions were collected for the German news platform Sport1 and form the dataset CLEF.

For many of the public datasets, the user interaction logs have already been split into sessions and, thus, included a unique session identifier. For some datasets, however, the "sessionization" had to be performed manually. In these cases, we applied a common heuristic and split the user logs after a certain period of inactivity [CMS99]. Following the default configuration of typical web servers, the logs were split after an idle time of 30 minutes.

2.4 Multi-Dimensional Comparison

After introducing the included techniques and the overall evaluation setup, this section now presents the main findings that were reported throughout the author's publications [JL17c], [KJL17], [LJ18b], and [LJ19b].

2.4.1 Accuracy Measures

In this section, the most important findings of the experiments in terms of the common accuracy measures are presented. Specifically, the outcomes for HR, MRR, Precision, Recall, and MAP at a list length of 20 will be reported.[13] Later on, the impact of the list length is discussed in more detail. In the following discussion, the tested techniques are furthermore classified into simple or baseline techniques like SKNN, CT, or AR that do not require an extensive training process, and complex methods that, in contrast, do.

Table 2.2: Accuracy results for a list length of 20 for two e-commerce datasets.[14]

(a) RSC15

RSC15					
Metrics	MAP@20	P@20	R@20	HR@20	MRR@20
NARM	**0.0357**	**0.0735**	**0.5109**	0.6751	0.3047
SMF	0.0352	0.0725	0.5065	**0.6761**	0.3071
STAMP	0.0344	0.0713	0.4979	0.6654	0.3033
V-SKNN	0.0341	0.0707	0.4937	0.6512	0.2872
GRU4REC	0.0334	0.0682	0.4837	0.6480	0.2826
SR	0.0332	0.0684	0.4853	0.6506	0.3010
AR	0.0325	0.0673	0.4760	0.6361	0.2894
SKNN	0.0318	0.0657	0.4658	0.5996	0.2620
CT	0.0316	0.0654	0.4710	0.6359	**0.3072**
NEXTITNET	–	–	–	–	–

(b) RETAILROCKET

RETAILROCKET					
Metrics	MAP@20	P@20	R@20	HR@20	MRR@20
SKNN	**0.0283**	**0.0532**	**0.4707**	**0.5788**	0.3370
V-SKNN	0.0278	0.0531	0.4632	0.5745	**0.3395**
GRU4REC	0.0272	0.0502	0.4559	0.5669	0.3237
SMF	0.0246	0.0467	0.4121	0.5091	0.2455
NARM	0.0239	0.0440	0.4072	0.5549	0.3196
STAMP	0.0229	0.0428	0.3922	0.4620	0.2527
AR	0.0205	0.0387	0.3533	0.4367	0.2407
SR	0.0194	0.0362	0.3359	0.4174	0.2453
NEXTITNET	0.0173	0.0320	0.3051	0.3779	0.2038
CT	0.0162	0.0308	0.2902	0.3632	0.2305

General Findings. First, the adapted factorization-based approaches were almost always the worst performing methods in terms of the accuracy (except for SMF). Even though being sequence-agnostic, BPR surprisingly often performed best among the factorization approaches. In general, these results show that techniques designed under the assumption of the existence of long-term user histories are not necessarily well suited for session-based recommendation. Due to their poor performance, those techniques were excluded from further experiments in [Lud+19], and will not further be considered in this discussion. All results alongside a discussion of them can be found in [LJ18b].

[13] A list length of 20 is commonly used and, thus, ensures good comparability.

[14] The best values are highlighted in bold font and the best result from another category of algorithms, simple or complex, is underlined. Significant differences according to a Student's t-test with Bonferroni correction are indicated by the dashed line ($\alpha = 0.05$). Due to computational limitations, the results for NEXTITNET could not be determined for one of the datasets.

Table 2.3: Accuracy results for a list length of 20 for two of the music datasets.[15]

(a) NOWPLAYING

NOWPLAYING					
Metrics	MAP@20	P@20	R@20	HR@20	MRR@20
V-SKNN	**0.0193**	**0.0664**	**0.1828**	**0.2534**	0.0810
SKNN	0.0186	0.0655	0.1809	0.2450	0.0687
AR	0.0166	0.0564	0.1544	0.2076	0.0710
SR	0.0133	0.0466	0.1366	0.2002	0.1052
NARM	0.0118	0.0463	0.1274	0.1849	0.0894
SMF	0.0117	0.0457	0.1350	0.2113	0.0935
GRU4REC	0.0116	0.0449	0.1361	0.2261	0.1076
STAMP	0.0111	0.0455	0.1245	0.1919	0.0897
CT	0.0065	0.0287	0.0893	0.1679	**0.1094**
NEXTITNET	–	–	–	–	–

(b) AOTM

AOTM					
Metrics	MAP@20	P@20	R@20	HR@20	MRR@20
SKNN	**0.0037**	**0.0139**	**0.0390**	**0.0417**	0.0054
V-SKNN	0.0032	0.0116	0.0312	0.0352	0.0057
AR	0.0018	0.0076	0.0200	0.0233	0.0059
SMF	0.0015	0.0064	0.0195	0.0148	0.0082
SR	0.0010	0.0047	0.0134	0.0186	0.0074
NARM	0.0009	0.0050	0.0146	0.0202	0.0088
CT	0.0006	0.0043	0.0126	0.0191	**0.0111**
NEXTITNET	0.0004	0.0024	0.0071	0.0139	0.0065
STAMP	0.0003	0.0020	0.0063	0.0128	0.0088
GRU4REC	0.0003	0.0020	0.0063	0.0130	0.0074

Table 2.4: Accuracy results for a list length of 20 for the news dataset (CLEF).[15]

CLEF				
Metrics	P@20	R@20	HR@20	MRR@20
GRU4REC	**0.072448**	**0.626009**	0.568499	0.219950
V-SKNN	0.068508	0.593304	0.775558	0.223575
SKNN	0.065616	0.577133	**0.778078**	0.218566
SMF	0.061677	0.526821	0.706117	**0.234424**
AR	0.058454	0.505773	0.665970	0.216195
SR	0.058275	0.501955	0.671878	0.222966

Table 2.2, Table 2.3, and Table 2.4 show representative results of the most important remaining algorithms and a selection of the analyzed datasets. The main findings in terms of the accuracy can be summarized as follows:

The results for the e-commerce, music, and news domain are mostly aligned. However, for each domain, there exist datasets that seem to be more *sequential* (e.g., RSC15 and NOWPLAYING) than others (e.g., DIGINETICA and AOTM). In the *sequential* datasets, items more often occur in the same order and, thus, more sequence-aware techniques like GRU4REC and SR, have an advantage over sequence-agnostic techniques like SKNN or AR.

The neural approaches GRU4REC, NARM, and STAMP are mostly among the best-performing methods with regard to the accuracy measures HR@20 and MRR@20. However, when considering all remaining items in a session as relevant, these techniques are often ranked lower, especially in the music domain. Among the neural techniques, no constantly superior algorithm can be identified. Although the authors of NARM and later STAMP reported performance increases over GRU4REC for RSC15, which could be reproduced in the conducted experiments, the order of the techniques is not homogeneous for the other datasets. NARM, however, is the most stable of the three approaches across all datasets. NEXTITNET mostly showed a poor performance compared to the other neural approaches in the investigated setup.

[15]The best values are again highlighted in bold font and the best result from another category of algorithms, simple or complex, is underlined. Significant differences are once more indicated by the dashed line.

The factorization-based SMF method mostly led to results comparable to GRU4REC. For a few less *sequential* datasets (AOTM and 8TRACKS), the approach was able to outperform GRU4REC consistently for all measures. For the news domain, it even led the best overall results in terms of the MRR.

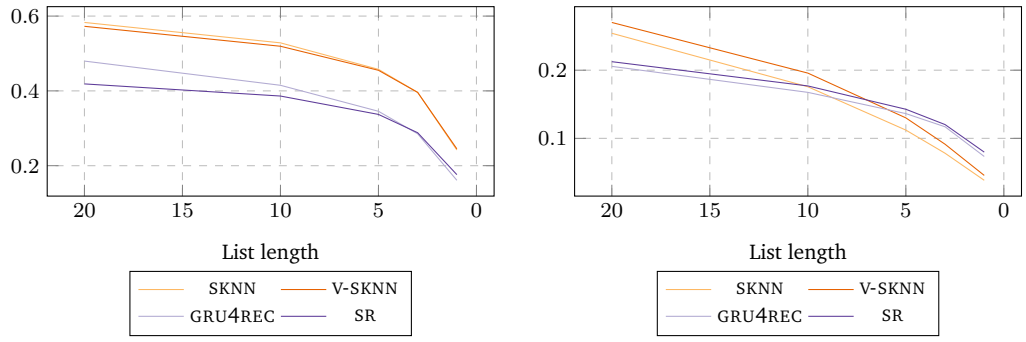
The simple pairwise frequent pattern techniques (AR, MC, SR) can mostly be found in the middle places in comparison. Usually, it is favorable to choose SR over AR and MC. Only for a few of the less *sequential* datasets (RETAILROCKET and AOTM), the *sequence-agnostic* AR approach performs slightly better. In general, however, SR is very competitive with the neural techniques and is sometimes even able to outperform them for the ranking-aware measure MRR at a list length of 20 (NOWPLAYING, 30MUSIC, and AOTM). The more sophisticated CT method can be seen as an improvement to SR and even achieves the best MRR results for most of the datasets, especially in the music domain.

Finally, the nearest-neighbor-based techniques presented by the author of this thesis in the included publications usually showed the best performance for most measures across all e-commerce and music datasets except for the RSC15 dataset, which seems to be particularly *sequential*. Only for the MRR@20 measure other techniques could outperform SKNN and V-SKNN, mostly CT. For the news domain, V-SKNN still showed a better performance in terms of the HR. In contrast to the other datasets, however, GRU4REC here interestingly performed better in terms of precision and recall.[16]

Overall, no consistent pattern in terms of the performance ranking can be identified. The results are very dependent on the measures and the dataset. The application of the simple and efficient AR and SR methods, however, can give an impression of the *sequentiality* of the data and, in consequence, serve as an indicator for the techniques that are most suitable for the situation. Furthermore, no consistent progress in terms of the predictive performance of neural techniques proposed over recent years can be found. In contrast, comparably simple techniques based on nearest-neighbors (SKNN, V-SKNN) or frequent patterns (CT) show a strong performance across all experiments and are still mostly able to outperform the more recent sophisticated model-based techniques. Similar observations were made in [DCJ19] for the scenario of top-n recommendation, where recent neural approaches were seldom able to outperform properly applied and tuned baseline techniques.

Impact of the List Length. In the previously reported results, the list length was fixed at a size of twenty. However, the length of the recommendation lists can have an impact on the accuracy measures and, thus, on the ranking of the tested techniques. As a consequence, an analysis of different list lengths was performed in [LJ18b]. Regarding the HR, for example, the dependencies visualized in Figure 2.7a

[16] Note that the news dataset was only used in [LJ18b] and the measurements did not include more recent methods as well as the MAP.



(a) RETAILROCKET **(b)** NOWPLAYING

Figure 2.7: Hit rate (HR) for two of the datasets when reducing the recommendation list length from 20 to 1.

and Figure 2.7b can be observed across all dataset. When decreasing the list length, usually, the discrepancy between sequence-agnostic and sequence-aware techniques becomes smaller (RETAILROCKET), or the sequence-aware techniques even start to show a better performance (NOWPLAYING). Similar patterns can be observed for the metric Precision. The ranking-aware measures are less prone to this effect.

2.4.2 Additional Quality Criteria

As discussed in Section 2.3, it is advisable to analyze the performance of a recommender system not only in terms of accuracy metrics. Hence, a number of additional quality factors were measured across all algorithms, datasets, and domains.

Popularity and Coverage. First, insights about the popularity bias and the catalog coverage (or aggregated diversity) of the tested techniques are discussed. Alongside very poor accuracy results, the factorization-based techniques generally led to a very low popularity bias and high coverage. As the bad accuracy results can, however, not be ignored completely, these results are not of great value in the discussion and are thus neglected. For the remaining techniques no consistent overall ranking of the algorithms across the datasets can be determined. However, the results indicate the following general observations:

- The CT method performs very differently from all other approaches. Compared to other techniques, it consistently focuses on very popular items, which also results in a low coverage of the item catalog.
- The neural network techniques tend to have a lower popularity bias than the SKNN and V-SKNN methods, especially GRU4REC, which almost always has the lowest popularity bias and the highest coverage. Among the neural networks, STAMP consistently has the highest popularity bias and also the lowest coverage.

- By design, the frequent pattern technique AR also often focuses on the recommendation of rather popular items. At the same time, however, the catalog coverage is still not among the worst performing techniques. SR, in comparison, generally recommends less popular items, and shows a higher coverage than AR. Mostly, the popularity tendency for AR and SR is still lower than for V-SKNN, while, however, the coverage also tends to by lower.

Computational Complexity. In terms of the practical application of session-based techniques, also the computational complexity can play an important role. In general, models based on neural networks tend to require a time-intensive training procedure. At the same time, these techniques often have a substantial number of hyper-parameters that must be optimized for individual application scenarios. Table 2.5, as an example, shows an overview of the running times for two of the datasets, RSC15 and ZALANDO.

Techniques like V-SKNN and SR do not rely on learning a model. In contrast, they only need a single iteration over the training interactions to initialize some data structures and thus have very short training times. Also the prediction times are low enough for practical applications. CT requires more time in the training phase but can still be initialized efficiently. The prediction time, however, increases strongly with the number of items in the dataset. While the measured time for RSC15 might still be acceptable, predicting for ZALANDO becomes very slow.

In terms of the neural networks, the training times are, as expected, much higher. Training GRU4REC with approximately one and two hours (RSC15 and ZALANDO) was comparably fast. STAMP and NARM were many times slower but could still be executed in a reasonable amount of time. Training NEXTITNET, however, already took over a day on a dataset of modest size. A extensive optimization of hyper-parameters, thus, becomes exceedingly difficult. Note furthermore that the training was performed with a state-of-the-art GPU and an execution on a CPU usually took

Table 2.5: Running times for two of the e-commerce datasets.

	Training		Predicting (ms)	
Algorithm	RSC15	ZALANDO	RSC15	ZALANDO
GRU4REC (on GPU)	0.89h	1.51h	8.81	30.06
STAMP (on GPU)	1.25h	7.61h	13.79	51.84
NARM (on GPU)	4.36h	12.99h	9.72	28.69
NEXTITNET (on GPU)	26.39h	–	8.98	–
SR (on CPU)	17.35s	21.37s	3.40	8.66
V-SKNN (on CPU)	10.71s	5.48s	16.42	26.00
CT (on CPU)	5.91m	2.10h	57.66	327.83

five to ten times longer. In general, besides the number of interactions, the training and prediction times also increase strongly with the number of items in the dataset. In terms of predicting, however, the neural networks by design are fast in general.

2.4.3 Alternative Evaluation Setups

Besides the standard evaluation setup described in Section 2.3, in [JL17c] and [LJ18b], the author of this thesis also investigated several alternative approaches to evaluate session-based recommendation techniques.

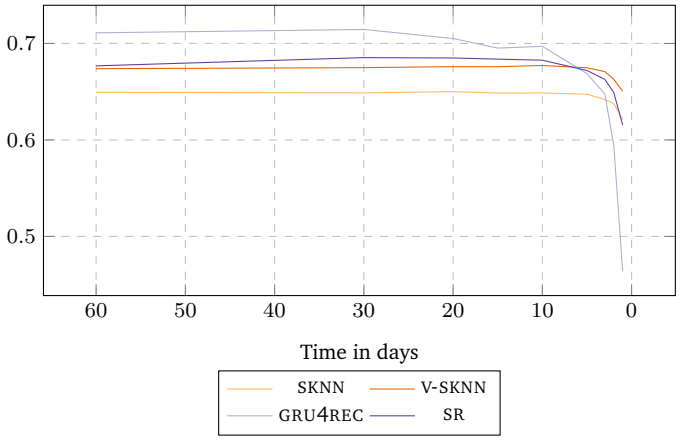


Figure 2.8: HR@20 for a single split of the RSC15 datasets when artificially reducing the size of the training set from 60 days to 1 day.

Sparsity Effects. First, the overall evaluation setup was slightly altered to investigate the effects of data sparsity on different techniques. While the test sessions are still treated as usual (incremental *given-N next-item prediction*), the training data was artificially reduced in its size from 60 days to only the last day. Figure 2.8, as an example, shows the results for the RSC15 dataset. The reduction of the training data surprisingly has almost no influence on the resulting performance. A similar pattern can be found for many of the datasets, at least in the e-commerce domain (see [LJ18b]). This observation also coincides with the results presented in [JLL17] that the user interactions are strongly influenced by recent trends in the data. Furthermore, it explains why sampling only recent user sessions in SKNN can even have beneficial effects (see Section 2.2.2).

Fixed-N Next-Item Prediction. In [JL17c], the author of this thesis also explored the *given-N next-item prediction* scheme with a fixed N. Specifically, only the first item in a session was revealed ($N = 1$) or only the very last item in each session was held-out in the testing phase ($N = |s| - 1$). Hereby, differences in the ability of the techniques to adapt to short and long sequences have been investigated. Table 2.6 shows the results for a single split of the RSC15 dataset. In general, SKNN outperforms

Table 2.6: Additional measurements for the RSC15 dataset.

Dataset	SECOND ($N = 1$)		LAST ($N = \|s\| - 1$)		PURCHASES	
Metric	HR@20	MRR@20	HR@20	MRR@20	HR@20	MRR@20
SKNN	0.716	0.355	0.446	0.196	0.758	0.290
GRU4REC	0.655	0.300	0.388	0.174	0.542	0.215

GRU4REC. Interestingly, both SKNN and GRU4REC show a stronger performance for the session beginning, while the differences between the techniques decrease when predicting only the last item of the session. Note however that these results are take from early experimentations that relied on the first version of GRU4REC, which did not include the improved loss functions [Hid+16a].

Predicting Purchase Events. In addition to the *fixed-n next-item prediction* scheme, in [JL17c], the author of this thesis evaluated the techniques SKNN and GRU4REC with an alternative protocol that includes multiple interaction types. Specifically, for the RSC15 dataset, besides item view events, purchases of items were additionally considered. It is a natural assumption that purchased items might be most relevant to the user in a session and, thus, form a reasonable ground truth. In general, the protocol follows the common incremental *given-N next-item prediction* but a prediction is only made and subsequently evaluated when the next-item refers to a purchase event. The input for the algorithms, however, still solely consists of interactions of the type *view*. Table 2.6 shows the results of this evaluation in the column *PURCHASES*. In contrast to the standard *given-N next-item prediction* scheme, the difference in terms of the HR as well as the MRR between SKNN and GRU4REC is even more substantial. Again, these experiments were conducted with the first version of GRU4REC [Hid+16a]. Furthermore, the architecture of GRU4REC is, of course, not designed and trained to predict the next purchases. The results, however, again show that the SKNN method is at least a very strong baseline approach for session-based recommendation scenarios.

2.5 Users' Perception of Session-based Recommendations

As mentioned in Section 1.3, session-based recommendation techniques are mostly evaluated in an *offline* experimental setting. The main findings presented in the previous section indicate that rather simple nearest-neighbor approaches like the SKNN or the V-SKNN method often perform at least as good or often even better than recently proposed complex models in terms of accuracy measures. The results, however, at the same time also show a stronger popularity bias and a lower coverage for such simple techniques. This ultimately leads to the following question: How do the techniques compare in a field test and how is the quality of the recommendations perceived by the users?

Hence, alongside another offline evaluation, the author of this thesis conducted an online user study to compare five different algorithms, both simple as well as complex ones [LJ19b]. Recent findings indicate that results are more reliable when the participants are actually able to consume the recommended item [Loe+18; JLJ15b]. Thus, the presented study simulated an interactive online radio station, in which the users received recommendations based on a start track that they chose. They could then actually listen to all recommended songs and give feedback, which furthermore an influence on the list of upcoming tracks. The algorithms were compared in terms of the number of liked or skipped tracks as well as through a post-task questionnaire.

In terms of previous works, the number of user studies on the perception of session-based recommendations is comparably low. Two related studies from the music domain can be identified with [BOL09] and [KJ17], in which different playlist continuation approaches are compared in terms of the perceived quality of the recommendations. Besides the music domain, various studies on the users' quality perception of recommendations have been conducted, for example, in the movie domain [Eks+14]. All studies, however, address a different setting or do not included recent session-based techniques.

Throughout the next sections, the study design is explained in more detail, the tested algorithms are discussed, and, finally, the results of both the *online* as well as the *offline* evaluation are presented and compared.

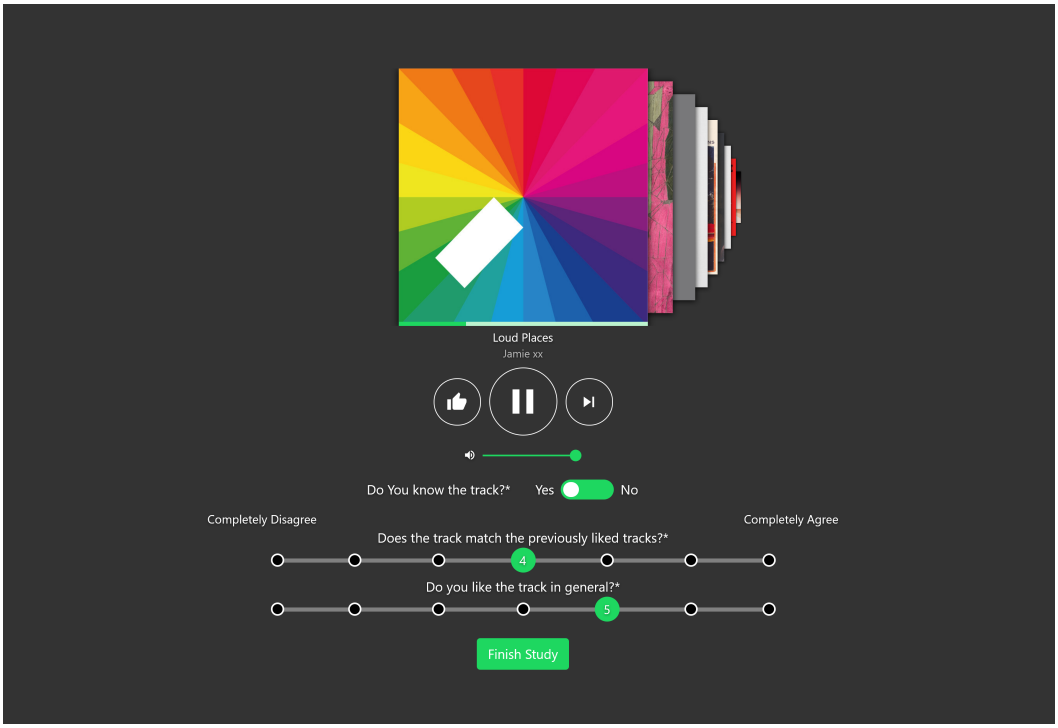


Figure 2.9: Interface of the interactive radio station.

2.5.1 Study Design

The main research goal was to understand how different algorithms affect the quality perception of users. Therefore, the author of this thesis developed an online radio station application to conduct a corresponding study. The task for the participants can be summarized in the following three steps:

1. The users were first provided with a search interface to find a pleasing track as the seed track for the radio station. At this, they could listen to 30-second excerpts of the tracks to simplify the decision-making process. Finally, the radio had to be started.

2. Next, the users were confronted with the main radio interface (see Figure 2.9). Again, the users could listen to a snippet of the song. Besides pausing the playback and skipping the song, they were able to give feedback in numerous ways. A “Thumbs-up” button could be pressed as an explicit statement of liking the current track. Positive feedback, furthermore, influenced the upcoming recommendations (visualized by the cover arts in the playback queue). Furthermore, the users were asked to respond to three basic questions; (a) if they know the track already, (b) if the track matches the previously liked tracks, and (c) if they like the track in general.

3. When the participants gave feedback to enough tracks (15 at minimum), they could continue to a post-task questionnaire consisting of 10 quality-related questions (7-point Likert scale). These questions are shown in Table 2.7 and were based on the general frameworks proposed in [Kni+12] and [PCH11]. Furthermore, one question had the purpose of checking the attention of the user (Q8). Another four questions were related to the participants' general taste in music and their music listening behavior.

Table 2.7: Questions about the users' quality perceptions.

Question	
Q1	I liked the automatically generated radio station.
Q2	The radio suited my general taste in music.
Q3	The tracks on the radio musically matched the track I selected in the beginning.
Q4	The radio was tailored to my preferences the more positive feedback I gave.
Q5	The radio was diversified in a good way.
Q6	The tracks on the radio surprised me.
Q7	I discovered some unknown tracks that I liked in the process.
Q8	I am participating in this study with care so I change this slider to two.
Q9	I would listen to the same radio station based on that track again.
Q10	I would use this system again, e.g., with a different first song.
Q11	I would recommend this radio station to a friend.
Q12	I would recommend this system to a friend.

2.5.2 Compared Techniques

The author of the thesis included five different techniques for the creation of the recommendations as a "treatment" and, furthermore, trained them on Spotify's Million Playlist Datasets (MPD), which was released in the course of the 2018 ACM RecSys Challenge [Che+18]. From the methods presented in Section 2.2, AR, V-SKNN, and GRU4REC were included. AR and V-SKNN already showed a strong performance for the MPD in [Lud+18], and GRU4REC was selected over the other neural techniques due to its better scalability. Furthermore, the approach *Collocated Artists - Greatest Hits* (CAGH) was implemented, which simply recommends the most popular tracks of similar artists. In [BJ14], this technique has proven to be successful in "playlisting" scenarios. Finally, we retrieved recommendations with the help of Spotify's API in real-time. Similarly for all techniques, a diversifying post-processing filter was applied to prevent that two tracks by the same artist are played subsequently. For further technical details, also regarding hyper-parameter optimizations and the offline evaluation, refer to [LJ19b].

2.5.3 Observations

Overall, the author of this theses recruited 316 participants over the crowdsourcing platform Mechanical Turk[17] ("Masters" only[18]) to collect 50 unique reliable submissions per treatment group.

Table 2.8: Statistics for the item-specific questions (mean and standard deviation).

Algorithm	Track known (%)	Thumb Up	Track matches the playlist	Like the track in general
AR	8.61	6.48 ±3.58	4.06 ±1.60	4.34 ±1.37
CAGH	10.83	5.38 ±3.77	5.15 ±1.14	5.03 ±1.22
GRU4REC	9.30	5.36 ±3.63	4.61 ±1.52	4.94 ±1.31
V-SKNN	10.13	5.63 ±3.13	5.31 ±1.04	4.94 ±1.06
SPOTIFY	7.00	4.48 ±2.90	4.72 ±1.13	4.69 ±1.07

Item-specific Results. When analyzing the item-specific answers and behavioral patterns of the participants reported in Table 2.8 the observations that are presented in the following can be made:

- Surprisingly, AR led to the significantly more like ("Thumbs-up") statements[19], although the participants stated that the recommended tracks matched the previous songs the least and they furthermore rated the track quality the worst.
- In terms of the characteristics of the recommended tracks, GRU4REC and, especially, SPOTIFY played the least popular tracks. The tracks suggested by SPOTIFY were, furthermore, significantly less often known by the users. Finally, the popularity of the tracks is highly correlated with the number of likes (0.89).
- V-SKNN and CAGH matched the previously played tracks significantly better than any of the other approaches.
- No specific differences could be determined regarding how much the users liked the recommended tracks in general.

Post-Task Questionnaire. When examining the users' answers to the post-task questionnaire, the following observations can be made:

- All methods received very positive feedback for all the questions and were seemingly able to satisfy the participants very well.

[17] `https://www.mturk.com`

[18] This option ensures a minimum average rating of the "workers" that participated in the study.

[19] In general, one-way ANOVA with a subsequent Tukey post-hoc test was used to test for significant differences, when the required pre-requisites were given. Otherwise, a Kruskal-Wallis with a subsequent Mann-Whitney-U test was applied.

- For V-SKNN, the participants attested that they liked the radio significantly better, and also preferred the matching quality over AR, GRU4REC, and SPOTIFY (Q1+3 in Table 2.7). Also, their likeliness to use the system again or even recommend it to a friend was significantly higher than for GRU4REC (Q10+12).

- SPOTIFY excelled in helping the users to discover unknown tracks and outperformed all other techniques significantly in that regard (Q7).

- CAGH, again, proved to be a very simple but nevertheless strong baseline in music recommendation scenarios.

Discrepancy to Offline Results. Finally, all techniques were also compared in an offline experimental setup. The results in terms of precision (P), recall (R), HR and MRR at a list length of five are presented in Table 2.9.[20] The evaluation scheme follows Section 2.3 and is described in [LJ19b] in more detail. The main findings in comparison to the *online* evaluation are:

Table 2.9: Offline results for the investigated techniques that were determined in a five-fold cross-validation on the MPD.

Algorithm	P@5	R@5	HR@5	MRR@5
V-SKNN	**0.271**	**0.044**	0.137	0.077
GRU4REC	0.161	0.028	**0.151**	**0.096**
AR	0.234	0.037	0.135	0.081
CAGH	0.172	0.024	0.052	0.026
SPOTIFY	0.009	0.001	0.002	0.001

- In line with the experiments in Section 2.4, the offline performance can depend on the chosen scheme and metrics. S-KNN shows a stronger predictive performance for the *given-n remaining item prediction*, while GRU4REC is the winner when predicting only the *next-item given-N*.

- As indicated by the *online* results, SPOTIFY, probably by design, recommends rather unpopular and novel tracks and, thus, shows a very limited offline performance in terms of accuracy measures.

- No clear correlation between offline and online results can be identified. However, with V-SKNN showing a better perceived quality in comparison to GRU4REC, precision and recall might be the more suitable offline indicators for online success in the given scenario.

[20] In contrast to the previous section, the list length was reduced as only a single recommended item is visible to the user in the online radio application.

Overall, the comparably simple nearest-neighbor technique V-SKNN could corroborate the strong offline performance in comparison to the more complex deep learning technique GRU4REC. The popularity bias, however, was also noticeable in this field test. However, it had no negative effects on the quality of the recommendations as perceived by the users that participated. Finally, the good perception of SPOTIFY's recommendations once more shows that bad offline results do not necessarily indicate a similarly poor real-world performance.

3 Exploring Session-Awareness in E-Commerce

As already discussed in Section 1.1, the problem of recommendation in the research literature is often abstracted to the matrix completion task, while neglecting practical limitations and challenges. With the work presented in [JL17a], [JLL17], and [JL17b], the author of this thesis focused on particular challenges of this problem formulation in the e-commerce domain. Here, the following problems that highly encourage session-awareness can be identified.

First, algorithms that are designed to solve the common matrix completion problem usually construct long-term user preference models. Customers of an e-commerce website, however, might visit the shop with a very specific intent in mind, e.g., to buy a swimming suit for an upcoming vacation. When solely relying on historical feedback from previous session, the preference model cannot consider the user's current intention. Thus, it is crucial for an e-commerce recommender system to be *session-aware* and adapt to such short-term goals of the user to provide valuable recommendations [SHB05; TB14; JLJ15a].

Second, the matrix completion task is designed to provide the user with recommendation of items that he is not yet aware of. However, from a business perspective, it might also be very valuable to recommend items that the user has previously shown interest in, e.g., in the ongoing or one of the more recent browsing sessions. The recommendation of such an item might then have a convincing effect and encourage the user in his intent of finally purchasing the item, or eventually of purchasing it again. Also from a user's perspective *session-aware reminders* can have the beneficial functionality of providing convenient navigation shortcuts to products that he or she might want to keep in mind (see [Sch+16]).

Besides these aspects of session-awareness, also other particularities of the e-commerce domain might have a big impact on the success of recommendations and are seldom covered in the academic recommender system literature. In the given session-aware context, the author of this thesis, furthermore, investigated the importance of short-term trends in the collective user behavior and the impact of price reductions on the suggested items. Some works indicate that recommending popular items in real-world scenarios is usually a safe and sometimes even good

strategy [Gar+14; GH16]. As many domains in e-commerce (e.g., the fashion domain) are subject to seasonal trends, explicitly accounting for short-term popularity trends might be beneficial. Furthermore, it is easily imaginable that the occurrence of a discounted item in a recommendation list might influence the user's choice, e.g., when three similar swimming suits are presented (in terms of quality and price) but for one of them the price is significant reduced.

Overall, not only the user's long-term preferences but a multitude of different short-term and session-related factors might influence the value of a recommendation given to the user. As such factors are, to a great extent, unexplored in the recommender system research literature, the author of this thesis systematically explored the importance of them and, furthermore, utilized them in designing novel session-aware recommendation approaches in two specific settings.

The first part of this chapter mainly focuses on the exploration of such aspects in the setting of fashion recommendation. Besides a statistical analysis of success factors, a number of simple and more complex recommendation techniques are be proposed and evaluated. The second part (Section 3.2), as mentioned in Section 1.4, explores the suitability of the previously introduced techniques for the related scenario of personalized search in e-commerce.

3.1 Success Factors in Session-Aware Fashion Recommendation

As stated in Section 1.4, the author of this thesis was provided with a large log dataset recorded by the popular European online fashion retailer Zalando[21]. An outstanding particularity of this dataset is that it includes extensive information about the interactions of users with recommendations in the context of item views. It contains both the recommendation lists that were shown to the users as well as the information on which of the recommendations they clicked. In combination with information about the purchases of users, the log data thus allows determining which recommendations were ultimately "successful".

In consequence, the author of this thesis performed an extensive exploratory as well as systematical analysis of the data to answer the question of what actually makes a recommendation successful in the given scenario of online fashion recommendation [JL17a]. A particular focus of the analysis are the previously introduced aspects, i.e., short-term intents in the user session, reminding, discounts and recent trends. Besides analyzing theses particularities of the domain, the work presented in

[21] http://www.zalando.com

[JL17a], furthermore, proposes a generic research framework on how to perform such an analysis in general and find important factors that indicate the success of recommendations in any domain.

After presenting the results of the analysis in Section 3.1.1, Section 3.1.2 is dedicated to investigating how the insights of the analysis can be utilized for the design of new recommendation techniques. Specifically, the author of this thesis proposed a two-stage approach based on session-based nearest-neighbors and a subsequent neural network. The technique jointly considers a multitude of the previously discussed factors that might be of importance. Furthermore, the approach was evaluated in comparison with a number of session-aware heuristics that have proven to be successful in the given scenario [JLJ15a; LJL16; JL17a].

3.1.1 Analysis of Success Factors

In this section the main findings of the dataset analysis are presented, both for the exploratory statistical as well as the systematical approach to assessing "success" factors of the recommendations. Note that, from here, a "successful" recommendation always corresponds to a recommendation that was first clicked and subsequently the suggested item was purchased in the course of one user session.

Dataset Characteristics. Before introducing the insights from the analysis, some important characteristics of the dataset are briefly introduce. The interaction logs from Zalando were collected over the period of one year and include mainly item view, add-to-cart, as well as purchase events of 3.5M users that interacted with over 400,000 items. For each item, some metadata was provided, specifically, the brand, the color, the category, and a price level[22].

Each item view interaction is associated with a recommendation list of three items that was presented to the user. To ensure subsequent item view events did not happen accidentally for one of the recommended items, a special recommendation click event indicates such interactions.

A "successful" recommendation was defined as an item suggestion that leads to a purchase. However, the dataset is very sparse, many users are one-time visitors, and removing users without a purchase reduces the number of users by ca. 80% (760,000 of 3.5M). For the analyses and experiments reported in Section 3.1.2 the users were furthermore categorized into *frequent*, *regular*, and *occasional* users to investigate the behavior of different user groups (see [JLL17] for more details).

[22]The level was defined by binning the prices of the items per category.

Exploratory Statistics

First, the author of this thesis calculated several descriptive statistics for the *frequent* users to quantify the importance of *session-awareness*, *reminders*, *discounts*, and *recent trends*. In general, the *frequent* users, on average, clicked on an item in every hundredth recommendation list (1%), 14% of the clicks led to an add-to-cart action, and 7% finally resulted in the purchase of an item (successful).

Importance of Short-Term Intents. The presence and importance of special session-intents can, e.g., be analyzed through the metadata of the inspected items. On average, the frequent users, e.g., only inspected items in 2.7 of over 330 categories (including sub-categories) per session, which strongly indicates the existence of a special goal in many sessions. As a consequence, the author of this thesis mainly analyzed the successfulness of recommendations when having the same attribute values as the currently viewed item (category, brand, color, and price level).

In general, the conversion rate for recommended items that have similar features was much higher. In terms of the brand, the differences were most significant with a 345% higher success rate for the same brand. This is to some extent expected, as the affinity with a brand is usually strong in the fashion domain. A similar price level (158%), the same category (158%), and the same color (77%) also improved the conversion rate substantially.

Effectiveness of Reminders. First, to be able to analyze the impact of reminders, it is important that Zalando's recommender system provides such suggestions after all. In fact, around 10% of the recommendations in the log data are reminders.

Secondly, even though the number of reminders is comparably low, it is interesting that still nearly half of the successful recommendations (44%) were already known to the customer from the ongoing or a previous session. This number does not guarantee that the recommendation ultimately induced the purchase. However, it demonstrates that this type of recommendation is, indeed, beneficial for the user, at least as a navigation shortcut [Pla+06; Sch+16].

Lastly, more than 50% of the products, on the contrary, were new to the user. This distribution shows that both types of recommendations have their justifications and a recommendation list, therefore, should be a balanced mix of both.

Impact of Discounts. As stated in the beginning, the hypothesis that discounts in a recommendation list might make items seem more attractive to users is not far-fetched. To this end, similarly to the analysis of the short-term intents, a comparison of the conversion rate for discounted and non-discounted items in the recommendation lists was performed.

As expected, while the overall conversion rate for regular priced items was at 0.45%, the success rate for the discounted items was 1800% higher (8.1%). Note, however, that discounted items stood out in the three-item lists, as the deployed recommender system on average only included one discounted item in the list. Nevertheless, these statistics are a very strong indicator for the proposed assumption and highly encourage the inclusion of discounted items in recommendation lists in the e-commerce domain.

A possible limitation of such type of recommendations might, however, be that users perceive the discounted items as some form of active advertisement by the shop provider and, thus, might be less pleased with the shopping experience.

Influence of Trends. When analyzing the popularity effects in the three-item recommendation lists, in 43% of the cases, the most popular item was chosen over the alternatives. Thus, a slight popularity tendency can be determined in general.

This effect, however, is much more noticeable when the popularity was determined for a limited period of time and, this way, should capture recent trends. The average daily popularity[23], e.g., of successful recommendations was three times higher than for all recommended items.

Overall, alongside many more descriptive statistics that have been calculated in the course of this exploratory analysis, the presented figures indicate the importance of every single factor for the creation of successful recommendation lists. Therefore, it furthermore seems quite promising to include such aspects in a session-aware recommender system in some form.

Structured Feature-Based Analysis

As a follow-up work to the exploratory analysis of the *frequent* user dataset, the author of this thesis explored ways of determining the relative importance of the four potential factors that might lead to a successful recommendation. Therefore, in [JLL17], he proposed a systematic procedure to quantify the impact of the different particularities in the log data. To this end, the logs were used to frame the research question of what makes a recommendation successful as a binary classification task. Each recommendation that was displayed to a user can be labeled as successful in case the item was subsequently purchased, or as unsuccessful otherwise. Furthermore, a multitude of features can be engineered that correspond to one of the previously investigated phenomena and which should indicate the

[23] More details on how the popularity values and all statistics in general were determined can be found in [JLL17].

successfulness of a certain recommended item. Finally, after collecting a dataset consisting of many features and the success label per recommendation, common techniques for determining the individual feature importance with regard to the label can be applied.

Feature Engineering. Overall, the author of this thesis engineered a total of 95 features corresponding to *short-term intents*, *reminders*, *discounts*, and *recent popularity*. The latter category is, e.g., covered by comparably simple features like the individual item popularity during the last 14 days. With a certain discount level, furthermore, a good predictor variable for this aspect is directly provided in the data. The importance of reminders is, e.g., modeled by the number of previous views of the item in the current session, or the distance in days to the last recorded view event. Last, in terms of *short-term intents*, for example, features like the ratio of clicks that refer to the same brand, category, color, or price level were engineered. A comprehensive list of all features can either be found attached in the appendix of [JLL17] or in an online appendix[24]. All 95 features were calculated for the frequent users ending up with 8,500 positive samples of successful recommendations. Note that due to a strong class imbalance (100 times more unsuccessful recommendations) a random down-sampling procedure was applied to balance the dataset.

Table 3.1: Results of the statistical feature weight analysis in terms of the Gain Ratio and a Chi Squared test.

(a) Gain Ratio

Feature	Weight
Viewed before?	0.319
Any discount granted?	0.274
Discount level	0.274
Distance to last view (in days)	0.251
Current popularity (day)	0.249
Distance to last view (in sessions)	0.232
Distance to first view (in days)	0.199
Distance to first view (in sessions)	0.194
Number of previous views	0.181
Current popularity (week)	0.138

(b) Chi Squared

Feature	Weight
Current popularity (day)	1.000
Distance to last view (in sessions)	0.624
Distance to last view (in days)	0.619
Current popularity (week)	0.610
Number of previous views	0.603
Distance to first view (in sessions)	0.598
Distance to first view (in days)	0.595
Viewed before?	0.590
Any discount granted?	0.569
Discount level	0.569

Feature Importance. Table 3.1 shows the ten most important features when applying the two popular feature selection techniques *Gain Ratio* and *Chi Squared* [MRS08]. Overall, four different approaches were applied to minimize the risk of misleading results. As shown in Table 3.1, however, the ranking of the most impactful features is very consistent. Although the order is not exactly similar, the same top ten features were determined.

[24] `http://dx.doi.org/10.17877/DE290R-18094`

In terms of the included types of features, the importance values perfectly confirm the assumptions made in the previous section. In the top ten list, features related to previous item views are very prominent. The recent popularity, furthermore, also seems to play a very important role. Finally, features regarding a potential discount and the level of the discount are also present in all lists. Merely the session-intent-related feature are not in the top-10 lists shown in Table 3.1. However, many of the subsequent features in the ranking are related to the brand and the category of the currently viewed item as well as their presence in the ongoing user session.

Overall, the previously described analysis revealed some indications which factors can ultimately contribute to the success of recommendations. Next, the following sections will investigate how these insights can be operationalized when designing new recommendation approaches.

3.1.2 Operationalizing the Success Factors

Based on the previous findings, the purpose of this section is the proposal of new session-aware recommendation algorithms that can successfully utilize the insights. In contrast to Chapter 2, where the task for the algorithms in the session-based abstraction was to predict the immediate next item click, the focus here shifts to purchase events. These are naturally present in the e-commerce domain, furthermore more meaningful than item view events, and are thus defined as the relevant "ground truth" in this scenario. The goal of the techniques that are proposed in the following therefore is to better predict the next purchase.

Two-Stage Approach

Previous works showed that it is highly important to consider both, the long-term history of the user as well as his short-term intents in the ongoing or the last few user sessions. In [JLJ15a], the authors, therefore, proposed a two-stage approach, which was adopted in this work:

1. A *baseline* algorithm is applied to generate a ranked candidate set of items. This technique can capture long-term preferences in the data. The application of just a session-based technique is, however, also possible and can even be beneficial in comparison to long-term models (see [JLJ15a]).
2. In the second stage, a *session-aware re-ranking* procedure is performed to adapt to short-term signals, both of the particular user (session-intent) and also the community (recent trends). The previously identified success factors are mainly considered through the second-stage approaches.

In this two-stage approach, they choice of the number of candidate items that are selected in the first stage can be crucial. Furthermore, it is important how these two individual stages are connected in terms of the final ranking. The item scores of the first stage could, e.g., be entirely neglected or adjusted in a weighted scheme. The proposed techniques rely on the latter variant and consider 200 candidate items from the first stage, which overall showed the best result in terms of the prediction accuracy in the evaluation presented in 3.1.2 (see [JLL17] for more details).

Baseline Recommenders. In terms of the baseline recommenders applied in the first stage, as in [JLJ15a], the author of this thesis relied on three algorithms that have already been introduced in Chapter 2 for the task of session-based recommendation. Specifically, BPR was included as a personalized long-term model, and AR as well as SKNN as session-aware approaches. However, with regard to the given scenario all techniques were subject to small alterations.

First, BPR was applied in the originally intended way. As long-term histories are available for all users, there was no need for approximating the latent user vector from the few available interactions in the current session (see 2.2.3).

Second, AR still relies on item co-occurrence patterns and computes association rules of size two for items that appeared together in the users' sessions. However, instead of basing the recommendations solely on the last item in the current ongoing user session, the approach considers all most recent item views. In contrast to the session-based next-item prediction scenario, this adjustment showed to be beneficial for the session-aware next-purchase prediction.

Third, as in [JLJ15a], the SKNN approach is not limited to only the latest ongoing user session. In addition, also previous sessions from the user history were considered. This procedure showed to be beneficial in previous works, as the user might pursue a certain goal over multiple sessions. Furthermore, when there are only very few interactions in the current session, the recommendations can thereby be based on a larger amount of reference items. Technically, the last N sessions are simply merged into a single session, which is then utilized to find similar sessions from the past.

Heuristic Short-Term Adaption. In [JL17a], the author of this thesis tested a number of simple heuristic approaches to incorporate the different success factors that were determined in the analysis.

- **Feature Matching (FM)**: Given a ranked set of items, FM ranks those items up that share similarities with the recently inspected items in the current sessions. The more attributes the item matches, e.g., brand, color, or category, the higher it will appear in the re-ranked list (see [JLJ15a] for technical details).

- **Interaction Recency (REMIND)**: In [LJL16], the author of this thesis investigated different approaches to reminding the user of already known items. Among other techniques, REMIND showed a solid performance by simply moving the items that the user recently inspected to the top of the recommendation list in reverse chronological order (newest first).

- **Recently Popular (TREND-N)**: Corresponding to the feature engineering process, TREND-N simply computes a normalized popularity score for items over the last n days. In the given scenario, the popularity in the last 24 hours led to the best ranking and, thus, TREND-N will be reported for $n = 1$ in the following.

- **Discount Promotion (DP)**: Finally, items are assigned to a discount level from zero to four. Thus, DP simply ranks the items with the highest discount at the top of the list. Items with a similar discount are, furthermore, ordered by the baseline ranking.

A further analysis of the features engineered in Section 3.1.1 showed that the success factors from the different categories (e.g., session-intent or reminding) are seldom correlated. Thus, it was furthermore promising to combine multiple heuristic strategies in a hybrid re-ranking approach (HR). The results for the best determined combination amongst a multitude of tested weighted and cascading hybrids is reported in Section 3.1.2.

Model-Based Short-Term Adaption. An severe disadvantage of the previously proposed heuristic approach is that a substantial amount of manual tuning is required to determine the best possible way of combining the techniques in a hybrid approach. The author of this thesis, thus, investigated a model-based short-term adaption that learns to re-rank the candidate items.

The proposed approach is very similar to the feature-based analysis from Section 3.1.1. Again, a binary classification problem was framed from the given log data. This time, however, the task was to predict if an item will be purchased by a user given his recent interactions in the ongoing and previous sessions. Even though the target variable was different, most of the features that were used to describe the success factors could directly be adopted. The overall recommendation scheme can be described as follows:

1. Again, a baseline algorithm is applied to compute an initial ranked list of candidate items.

2. For each of these items, a vector of features is calculated given the current and previous sessions of the user, e.g., the fraction of item of the same brand in the current session.

3. A pre-trained model is applied to determine the probability that each specific item will be purchased given the generated feature vectors. The ranking is, finally, performed based on the probabilities predicted by the model.

Due to the modifications to the task, some of the features that have been introduced in Section 3.1.1, could consequentially not be determined anymore, specifically, all features that are related to the currently viewed item and to the recommendation list. In this session-aware context, the number of features, therefore, was reduced from 95 to 32. A schematic overview of all features that were finally employed can be found in the included publications [LJ18b].

Not only the feature engineering procedure was adjusted but also the generation of a training dataset. A positive example in this problem definition referred to an item view event that led to a purchase in the same or a subsequent session of a user. On the contrary, a non-purchased item inspection formed a negative example. To create a balanced dataset of examples, for each purchase, a non-purchased item was randomly sampled from the same session.

Given a labeled training dataset, the author of this thesis next tested various machine learning approaches in a ten-fold cross-validation procedure, e.g., logictic regression, decision trees, random forests, or feed-forward neural networks, to determine the best-performing approaches and suitable hyper-parameters. Finally, feed-forward neural networks led to the best classification performance in terms of the accuracy. The second best techniques could be found in Random forests. Both approaches, namely DEEPRANK and RFRANK, were then trained on the full dataset and employed in the second stage for the session-aware re-ranking. More technical details are provided in [JLL17] or the appendix, respectively.

Evaluation Scheme

Overall, the strategies and baseline recommenders have been evaluated for the three user groups in the Zalando dataset. Besides the frequent users, the approaches were also tested for the regular and occasional user (see [JLL17]).

The evaluation protocol followed [JLJ15a] and can be classified as an alteration of the incremental *given-N next-item prediction*. Instead of predicting the next item view in an ongoing user session, the next purchase had to be predicted by the tested algorithms. Thus, successively each purchase event in the test sessions was evaluated, while the algorithms had access to the previous item view interactions of the user, both in the current and the preceding sessions. Following [JLJ15a], the number of previous sessions available for a short-term adaption was limited to 6.

Table 3.2: HR@10 and MRR@10 results for Zalando's *frequent* users.

Baseline	SKNN		AR		BPR	
Metric@10	HR	MRR	HR	MRR	HR	MRR
No re-ranking	0.268	0.091	0.123	0.046	0.062	0.021
FM	0.281	0.093	0.145	0.052	0.119	0.046
REMIND +FM	0.306	0.097	0.266	0.096	0.262	0.111
DP +FM	0.316	0.177	0.242	0.120	0.168	0.094
TREND-N +FM	0.361	0.187	0.233	0.103	0.216	0.096
RFRANK	0.381	0.248	0.274	0.150	0.241	0.119
HR (TREND-N,DP)+FM	0.382	0.220	0.262	0.121	0.225	0.100
DEEPRANK	**0.405**	**0.284**	0.322	0.205	0.301	0.188

In contrast to the evaluation scheme in Chapter 2, in session-aware recommendation long-term user histories have to be available to the algorithms. Therefore, splitting the dataset can not be performed solely time-based anymore. Referring to the terminology introduced in Section 1.3, the dataset was split *user-wise* at *community-level*. Each user history was then divided into a training and a test part, *time-based* at *session-level*. In fact, the most recent 20% of the users' sessions formed the test set, while the rest was used to train the models. To account for random effects in the results, the evaluation was performed in a user-wise five-fold cross-validation scheme, i.e., each fold consisted of the full histories of 80% of the users.

In terms of the measured accuracy metrics, the evaluation setup follows the session-based next-item prediction scheme. Besides the HR, the ranking-aware MRR was recorded, in this case for a list length of 10 (again, following [JLJ15a]).

Results

As an example, the results for the frequent users are presented in Table 3.2. Although the results for other user groups are different in terms of the range of the values, they, however, show an identical ranking of the tested techniques and are provided in the included publications (see [JLL17]).

Baseline Performance. In line with the results presented in Section 2.4 for the next-item prediction task, also in the examined session-aware next-purchase setting the SKNN method shows a superior performance in comparison to AR and BPR. Interestingly, not even the presence of long-term user histories enables BPR to achieve competitive results. This fact, again, emphasizes the importance of considering the recent user interactions in an e-commerce recommender system.

Heuristics. All proposed heuristic re-ranking approaches constantly lead to accuracy improvements over the baselines. The feature matching technique FM, furthermore, also consistently helps in combination with any other heuristic approach and was, thus, always applied as a pre-processing step to the re-ranking, i.e., items without any match were excluded from the set of candidates. In comparison to REMIND and DP, TREND-N usually improves the baseline the most, which to some extent is consistent with the feature importance values presented in Section 3.1.1 (see also [JLL17]). Finally, a weighted hybrid, as expected, can constantly improve the performance further by incorporating all different factors.

Model-Based Approach. With the model-based approach that was proposed by the author of this thesis, all success signals discussed in Section 3.1.1 can be leverage in a more dynamic and sophisticated way. While RFRANK only outperforms the fine-tuned hybrid approach HR in terms of the MRR, DEEPRANK shows a superior performance across almost all user groups and metrics.[25] Especially in terms of the ranking-aware MRR metric the differences stand out.

Overall, the results of the conducted experiments indicate that it is highly important to generate recommendations with short-term signals in mind. Furthermore, incorporating a multitude of factors like user intents, reminders, and community trends in simple heuristic approaches already helps to improve the accuracy of session-aware e-commerce recommendations. Finally, modeling these factors more elegantly and applying modern machine learning techniques, particularly neural networks, further helped to improve the recommendation performance.

[25] The differences have, furthermore, been statistically significant according to a Student's t-test with Bonferroni correction ($p < 0.05$)

3.2 Session-Aware Personalized Search

Chapter 2 as well as the previous section mostly focused on providing automated recommendations in the context of an item that the user is currently inspecting. This context, however, is only one of many possible scenarios, in which personalized item suggestions can help the user to better explore the space of options. Other scenarios might, for example, be the landing page or the shopping cart. Moreover, an obvious functionality in the e-commerce domain is the catalog search tool.

In the latter scenario, common e-commerce sites usually provide the user with a number of pre-defined sorting options, e.g., “by popularity” , “by rating”, “by sales”, or by “by freshness”. Sometimes the option “by relevance” is offered, which might refer to the relevance for the particular user.

However, many e-commerce sites do not include a sorting option that explicitly refers to the user’s personal preferences, although, intuitively, the past shopping behavior could help to improve the ranking of the search results. They, e.g., could promote the user’s favorite brands, items that match the user’s price preferences, or items that the user showed interest in before.

In consequence, the author of this thesis investigated the value of personalizing search results with a case study in an e-commerce setting [JL17b]. Generally, as mentioned in Section 1.4, the task of personalizing search results given user activity logs is quite similar to the scenario of session-aware or session-based recommendation. The main difference is the search query, which is explicitly entered by the user. Assuming that a given generic search component can filter the entire item catalog based on a query, the setting can be reduced to the session-aware recommendation problem with a restricted catalog of items to recommend from.

Technically, the author thus mainly explored the application of several recommendation techniques and domain-specific heuristics for session-aware recommendation that have proven to be successful throughout previous works (see Section 3.1.2). Additionally, a number of techniques from the broader field of personalized web search were included. Finally, the evaluation of all techniques was performed in an offline experimental setting with log data provided by a popular German online retailer that is specialized in products for infants and small children.

In academia, only a few recent works can be identified that target the analysis of search log data in the e-commerce domain. In [Dua+13], [Liu+14], or [Yu+14], for example, the focus however lies not on the personalization of search results but rather on an unpersonalized diversification. The authors of [PS11] presented an

approach to personalizing the ranking of the results. In their approach, however, users manually had to adjust the ranking by assigning weights to different sort criteria with the help of multiple sliders.

In the following, first the general research methodology and the session-aware evaluation scheme are introduced. After describing the investigated algorithms for search personalization from both worlds recommendation and web search, finally, the main findings are presented.

3.2.1 Research Setup

One of the most important aspects regarding the conducted search personalization experiments is the procedure of defining the evaluation protocol and the relevance of items given the provided interaction logs. After providing some additional information about the dataset, this section will discuss the definition of the "ground truth" and, in consequence, how a structured evaluation was performed.

Dataset. The data used in the following experiments consists of anonymized behavioral user logs collected by the online retailer over a period of one month. Each time-stamped user event belongs to one of the following action types: item view, category browsing, search (query included), add-to-cart, or checkout. Furthermore, some basic metadata was provided for the items, e.g., the category, the brand, and a textual description. Finally, through web crawling the author of this thesis additionally obtained the search results for each included query and every possible sorting option that was offered by the shop at that time.

Ground Truth. Differently from the previous sections, the ground truth that the algorithms should predict can not simply be defined as the next clicked item, all remaining items, or the finally purchased products in a user session anymore. Given a search event, an item can only be relevant when it occurred in the search results. Potentially, an item that was viewed subsequently in the same user session could be seen as relevant. However, subsequent clicks might be biased by the order in which the results were originally presented to the user. Thus, items in the results were defined as relevant only in case they were later on also purchased by the user in the same browsing session.

Evaluation Protocol. The evaluation protocol applied in this personalized search scenario shares many similarities with the protocol presented in Section 3.1.2. In general, the dataset again was split *user-wise* at *community-level*. For each user, the most recent 20% of the sessions were selected for testing (*time-based* at *session-level*).

Sessions in the test set were then evaluated similarly to the incremental *given-N prediction* scheme. However, as in [MR11], only search events are considered as evaluation points, and items defined as relevant for each specific search event form the ground truth. In terms of accuracy metrics, the HR and the MRR were measured for a list length of ten[26].

The number of results might vary strongly for different search queries and can have an substantial impact on the ranking accuracy. Re-ranking, for example, a set of only ten candidate items leaves less margin for improvements than for a longer result list, e.g., of 50 items. Thus, the evaluation was performed with multiple thresholds regarding the minimum number of items in the search results, specifically 5, 10, 20, and 50. Finally, the tests were performed in a five-fold user-wise cross-validation procedure to account for random effects.

More specific details, e.g., on the dataset (including statistical characteristics), on the crawling process, or regarding the splitting procedure can be found in [JL17b] or the appendix, respectively.

3.2.2 Compared Algorithms

The compared algorithms mostly include recommendation strategies that were previously introduced in Section 2.2.2 and 3.1.2. Specifically, BPR, SKNN, FM, TREND-N, and REMIND were employed to capture signals that have proved to be valuable in session-aware recommendation throughout previous works [JL17a; JLL17].

In the field of web search personalization, a customization of the search results is typically achieved by collecting or learning short- and long-term user preferences as well as modeling the context of a search action from various types of information, e.g., based on document term vectors, ontologies, language models, or different machine learning techniques [TSZ06; SMB07; BSD10; MR11; Ben+15]. Like the FM method, some works rely on additional metadata besides the textual information of the webpages, e.g., the user's current location [Ben+11], or information from social media [ZLW12b; Car+09]. Mainly the following types of personalization settings can be found in the literature: Techniques that (a) directly extend the search algorithm with the integration of a personalization component [HKJ03], (b) expand queries in a personalized manner [CFN07; ZLW12a], and (c) re-rank search results by applying a post-processing strategy [DSW07; MR11].

[26] Assuming a standard screen resolution of 1920 by 1080, a user could see around 10 search results on the first result page at the same time when using the online shop.

As the latter type of techniques perfectly suits the given scenario, PCLICK (see [DSW07]) and the content-based approach (CB) presented in [MR11] were included in the experimentation. Other approaches could not be adapted due to the particularities of the setup. PCLICK has some similarities with REMIND and follows the idea that users often search for the same or a similar item again. Thus, items that the users already showed interest in before for a similar or the same search query are ranked up in the result list. The more often an item was clicked, the higher it is ranked. In contrast, CB relies on content information. Specifically, as the method originates from web search, it bases the ranking on the text of a website, which, in the given scenario, corresponds to the textual item description. This information is represented by TF-IDF vectors and then again used to represent the user preferences. A mean vector is calculated from the items he or she interacted with. For each user, a short-term as well as a long-term preference vector is determined and, furthermore, combined in a weighted sum according to a specified hyper-parameter. Finally, the results are ranked by calculating the cosine similarity between the final user preference vector and the TF-IDF representation of a candidate item.

The included recommendation techniques were, furthermore, combined in three weighted hybrid approaches (HR). First, an entirely heuristic one that was based on TREND-N and the session-aware FM approach. Thus, no long-term user model had to be trained. Second, this hybrid was extended by including BPR. Finally, REMIND was incorporated to include all factors in the recommendation process, long-term preferences, trends, reminders, and session-aware context information.

The weights of the hybrid combinations of techniques as well as all other important algorithm hyper-parameters were optimized in a grid search procedure and are reported in the included publications [JL17b].

3.2.3 Findings

The results of the experiments in all configurations are presented in Table 3.3. In terms of the minimum number of items found for a query, the results are very consistent across all thresholds. However, the HR and MRR values naturally decrease to some extent with an increasing number of items to rank. Overall, no technique performed worse than the best initial sorting provided by the online shop itself ("by sales") and the main findings are the following:

- Regarding the web search personalization techniques, PCLICK was is slightly better than sorting "by sales", which, however, can be attributed to the small number of users in the data that repeated a query over the short data collection period of only one month. In contrast, CB shows a solid performance with results even slightly better than those of the usually promising SKNN method.

Table 3.3: *HR@10* and *MRR@10* results for 5,000 frequent users. The best values are printed in bold. Differences between the best performing method and the second best that are statistically significant according to a Wilcoxon signed-rank test ($\alpha = 0.05$) are highlighted by the dashed lines.

Min. nb. of result items	10		20		50	
Metric@10	HR	MRR	HR	MRR	HR	MRR
HR (TREND-N,FM,BPR,REMIND)	**0.675**	**0.382**	**0.636**	**0.363**	**0.584**	**0.324**
REMIND	0.652	0.371	0.619	0.352	0.572	**0.324**
HR (TREND-N,FM,BPR)	0.624	0.304	0.565	0.282	0.511	0.241
HR (TREND-N,FM)	0.604	0.289	0.564	0.267	0.504	0.235
BPR	0.579	0.284	0.535	0.262	0.477	0.226
FM	0.580	0.265	0.536	0.242	0.475	0.209
CB	0.535	0.258	0.487	0.238	0.416	0.198
SKNN	0.524	0.246	0.480	0.227	0.403	0.185
TREND-N	0.506	0.208	0.455	0.185	0.375	0.148
PCLICK	0.438	0.172	0.385	0.150	0.321	0.127
Shop baseline ("by sales")	0.433	0.168	0.380	0.147	0.314	0.125

- Surprisingly, in contrast to the results presented in 3.1.2, the long-term model BPR shows a stronger performance than the session-aware SKNN for search personalization. The comparably worse performance of SKNN might be explained with the position, in which search events usually occur in the user sessions. Often, users start the session by searching for an item in the catalog. At that time, however, no previous item view events are available and, thus, SKNN is limited in finding appropriate neighboring sessions.

- The simple session-aware heuristics FM and especially REMIND lead to very strong accuracy results. While FM is on par with BPR, REMIND outperforms all other techniques significantly. However, only reminding the users of known products might not be desired in practical applications. The strategy should, thus, rather be combined with alternative approaches that create more "surprising" recommendations. With this in mind, the strong performance of REMIND, again, indicates that relying solely on accuracy metrics might be misleading in terms of the online performance of a recommender system in the given scenario.

- The investigated hybrid approaches show that each of the included individual techniques can contribute to the creation of successful recommendations, at least to some extent. Already a combination of the simple TREND-N and FM methods leads to good results, which are again improved by including long-term preferences (BPR). When, furthermore, REMIND is added to the hybrid approach, the techniques jointly outperform all other tested approaches significantly. The final weights that were determined in an extensive grid search procedure, furthermore, emphasize the importance of the different techniques or factors respectively (0.45 for REMIND, 0.4 for TREND-N, and only 0.075 for BPR and FM).

Overall, despite being applied in a different personalization scenario, simple session-aware techniques again prove to be very successful. The results, furthermore, show that a multitude of different factors is again important for the creation of suitable item suggestions, not only in item-item-recommendation but also in the context of search personalization. A hybrid approach that considers long-term preferences, content-based contextualization, reminding, and also recent popularity trends leads to the best results in re-ranking the candidate items.

4 Summary & Conclusions

This thesis addresses a number of open research questions in the fields of session-aware and session-based recommendation that are presented in Section 1.4. In this concluding chapter, the main findings regarding both scenarios are summarized alongside a discussion of possible limitations of the research as well as future directions for follow-up works.

Being able to provide suitable item recommendations solely based on a few recent interactions in an ongoing user session is a highly relevant problem in practical scenarios. The session-based problem setting, thus, enjoys growing interest in the recommender system research community and several algorithmic approaches were proposed over recent years. However, no common standard evaluation procedure has been established so far, i.e., different datasets, data preprocessing schemes, evaluation protocols, baseline techniques, and metrics are employed throughout the presented works. This thesis and the included publications, therefore, propose a publicly available framework for researchers, which includes many recent techniques, ships with a number of datasets from different domains, and, thereby, enables future works to be better comparable and, more importantly, reproducible.

Moreover, this thesis provides an extensive overview of session-based recommendation techniques, which ranged from simple nearest-neighbor techniques, over factorization-based approaches, to the most recently proposed deep learning models. Furthermore, all these algorithms were tested in an extensive multi-dimensional comparison in a fair manner. Surprisingly, the experiments revealed that, in most cases, a simple nearest-neighbor technique outperforms modern approaches based on neural networks across many domains, datasets, and metrics. At the same time, the computational complexity is much lower. The better performance, however, often comes with a slightly higher popularity bias and a lower catalog coverage, which might be undesirable in a practical application. A user study in the music domain, however, shows that, at least in the investigated scenario, this particularity has no negative impact and the nearest-neighbor technique can provide recommendations that were perceived also of higher quality by the users.

New neural network architectures for session-based recommendation are constantly published and report improvements over previous models for specific setups. Apart from the fact that stronger baseline techniques like the nearest-neighbor methods are seldom considered, the extensive experiments, furthermore, show that the improvements over previous models often do not transfer to other domains and datasets. In fact, in several cases, the earliest neural network GRU4REC still performs best. In future works, new techniques, thus, will be integrated into the proposed framework to better asses the ongoing progress regarding session-based techniques.

Further ideas for future works include exploring session-based recommendation with multiple interaction types or additional metadata, providing a standard setup for these scenarios, and extending the session-based techniques to the session-aware scenario by including long-term histories. Furthermore, session-based models are usually trained to predict the immediate next item in a session. As, however, all remaining items in a session might be of relevance to the user, future models should also include these items in the training procedure. Finally, as the performance of different algorithms sometimes varies strongly across different metrics, domains and datasets, more research is required to determine why particular algorithms work better in particular scenarios or for specific datasets.

In terms of the session-aware recommendation scenario, the main intuition behind the covered research questions was to provide a better understanding of what makes users click on the provided recommendations in an online fashion shop. The author of this thesis performed a systematic analysis of the characteristics of successful real-world item suggestions, which has previously not been done for the e-commerce domain. The analysis shows that several aspects regarding the most recent user sessions are of particular importance. Specifically, reminding the user of previously inspected clothes and matching the recommendations with the most recently inspected items, e.g., recommending from the same category, showing clothes in the same color, or focusing on a specific brand leads to substantially more purchases. Furthermore, discounts and considering recent trends in the community have a beneficial effect on the conversion rate.

These insights were furthermore utilized in a new approach that jointly considers the success factors in a sophisticated way and, in consequence, provides more accurate recommendations than previous methods. The approach operates in a two-stage scheme. First, BPR is employed to consider the users' long-term preferences and, second, a feed-forward neural network effectively re-ranks a certain number of suitable items with respect to particularities of the most recent user sessions. Moreover, additional experiments show that similar aspects also help in the related scenario of session-aware search personalization in e-commerce.

While the analyzed and determined success factors are very specific to the given application domain of fashion recommendation, the procedure of determining the importance of any imaginable factor functions as a generic framework. Consequently, future works include the application of this framework to other domains, both from the e-commerce landscape as well as in completely different application scenarios. This way, it should be assessed if similar aspects are also success factors of recommendations in other settings. Similarly, the proposed DEEPRANK approach could be transferred to and tested in other domains.

Moreover, the proposed approach includes reminders and recently very popular items in the recommendation lists. Both types of item suggestions might not contribute to a satisfactory user experience in many cases. Although including such recommendations is common practice, more research is required to find an optimal balance and to asses when and for whom to include these types of recommendations.

Overall, this thesis contributes to a better understanding concerning the importance of considering the users' recent sessions in the recommendation process and, furthermore, leads to implications on how to utilize them effectively across multiple domains and recommendation scenarios.

Bibliography

[AK12] Gediminas Adomavicius and YoungOk Kwon. “Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques”. In: *IEEE Transactions on Knowledge and Data Engineering* 24.5 (May 2012), pp. 896–911 (cit. on p. 38).

[Bae+15] Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. “Predicting The Next App That You Are Going To Use”. In: *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*. WSDM ’15. 2015, pp. 285–294 (cit. on pp. 7, 36).

[BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate”. In: *Proceedings of the 3rd International Conference on Learning Representations*. ICLR ’15. 2015 (cit. on pp. 32, 35).

[Ben+11] Paul N. Bennett, Filip Radlinski, Ryen W. White, and Emine Yilmaz. “Inferring and Using Location Metadata to Personalize Web Search”. In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’11. 2011, pp. 135–144 (cit. on p. 67).

[Ben+15] Paul N. Bennett, Kevyn Collins-Thompson, Diane Kelly, Ryen W. White, and Yi Zhang. “Overview of the Special Issue on Contextual Search and Recommendation”. In: *ACM Transactions on Information Systems* 33.1 (Mar. 2015), 1e:1–1e:7 (cit. on p. 67).

[BJ14] Geoffray Bonnin and Dietmar Jannach. “Automated Generation of Music Playlists: Survey and Experiments”. In: *ACM Computing Surveys* 47.2 (Nov. 2014), 26:1–26:35 (cit. on pp. 25, 34, 49).

[BK17] Veronika Bogina and Tsvi Kuflik. “Incorporating Dwell Time in Session-Based Recommendations with Recurrent Neural Networks”. In: *Proceedings of the 1st Workshop on Temporal Reasoning in Recommender Systems co-located with 11th International Conference on Recommender Systems*. RecTemp ’17. 2017, pp. 57–59 (cit. on p. 35).

[BOL09] Luke Barrington, Reid Oda, and Gert R. G. Lanckriet. “Smarter than Genius? Human Evaluation of Music Recommender Systems”. In: *Proceedings of the 10th International Society for Music Information Retrieval Conference*. ISMIR ’09. 2009, pp. 357–362 (cit. on pp. 10, 47).

[BSD10] Paul N. Bennett, Krysta Svore, and Susan T. Dumais. “Classification-enhanced Ranking”. In: *Proceedings of the 19th International Conference on World Wide Web*. WWW ’10. 2010, pp. 111–120 (cit. on p. 67).

[Car+09] David Carmel, Naama Zwerdling, Ido Guy, Shila Ofek-Koifman, Nadav Har'el, Inbal Ronen, et al. "Personalized Social Search Based on the User's Social Network". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. CIKM '09. 2009, pp. 1227–1236 (cit. on p. 67).

[CFN07] Paul - Alexandru Chirita, Claudiu S. Firan, and Wolfgang Nejdl. "Personalized Query Expansion for the Web". In: *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '07. 2007, pp. 7–14 (cit. on p. 67).

[CGT12] Paolo Cremonesi, Franca Garzotto, and Roberto Turrin. "Investigating the Persuasion Potential of Recommender Systems from a Quality Perspective: An Empirical Study". In: *ACM Transactions on Interactive Intelligent Systems* 2.2 (June 2012), 11:1–11:41 (cit. on p. 1).

[Che+12] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. "Playlist Prediction via Metric Embedding". In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '12. 2012, pp. 714–722 (cit. on p. 36).

[Che+18] Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. "Recsys Challenge 2018: Automatic Music Playlist Continuation". In: *Proceedings of the 12th ACM Conference on Recommender Systems*. RecSys '18. 2018, pp. 527–528 (cit. on p. 49).

[CMS99] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. "Data Preparation for Mining World Wide Web Browsing Patterns". In: *Knowledge and Information Systems* 1.1 (Feb. 1999), pp. 5–32 (cit. on p. 39).

[CXJ13] Shuo Chen, Jiexun Xu, and Thorsten Joachims. "Multi-space Probabilistic Sequence Modeling". In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '13. 2013, pp. 865–873 (cit. on p. 36).

[Dav+10] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, et al. "The YouTube Video Recommendation System". In: *Proceedings of the 4th ACM Conference on Recommender Systems*. RecSys '10. 2010, pp. 293–296 (cit. on p. 24).

[DCJ19] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. "Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches". In: *Proceedings of the 13th ACM Conference on Recommender Systems*. RecSys '19. 2019 (cit. on p. 42).

[Dju+14] Nemanja Djuric, Vladan Radosavljevic, Mihajlo Grbovic, and Narayan Bhamidipati. "Hidden Conditional Random Fields with Deep User Embeddings for Ad Targeting". In: *Proceedings of the 2014 IEEE International Conference on Data Mining*. ICDM '14. 2014, pp. 779–784 (cit. on p. 36).

[DSW07] Zhicheng Dou, Ruihua Song, and Ji-Rong Wen. "A Large-scale Evaluation and Analysis of Personalized Search Strategies". In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. 2007, pp. 581–590 (cit. on pp. 67, 68).

[Du+16] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. “Recurrent Marked Temporal Point Processes: Embedding Event History to Vector”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’16. 2016, pp. 1555–1564 (cit. on p. 36).

[Dua+13] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. “A Probabilistic Mixture Model for Mining and Analyzing Product Search Log”. In: *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. CIKM ’13. 2013, pp. 2179–2188 (cit. on p. 65).

[Eks+14] Michael D. Ekstrand, F. Maxwell Harper, Martijn C. Willemsen, and Joseph A. Konstan. “User Perception of Differences in Recommender Algorithms”. In: *Proceedings of the 8th ACM Conference on Recommender Systems*. RecSys ’14. 2014, pp. 161–168 (cit. on pp. 10, 47).

[Fen+15] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. “Personalized Ranking Metric Embedding for Next New POI Recommendation”. In: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI ’15. 2015, pp. 2069–2075 (cit. on p. 36).

[Gar+14] Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, and Amr Huber. “Offline and Online Evaluation of News Recommender Systems at Swissinfo.Ch”. In: *Proceedings of the 8th ACM Conference on Recommender Systems*. RecSys ’14. 2014, pp. 169–176 (cit. on pp. 10, 12, 19, 54).

[Gar+19] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam Shroff. “Sequence and Time Aware Neighborhood for Session-based Recommendations: STAN”. In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’19. 2019, pp. 1069–1072 (cit. on p. 35).

[GDF13] Florent Garcin, Christos Dimitrakakis, and Boi Faltings. “Personalized News Recommendation with Context Trees”. In: *Proceedings of the 7th ACM Conference on Recommender Systems*. RecSys ’13. 2013, pp. 105–112 (cit. on pp. 7, 36).

[GDJ10] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. “Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity”. In: *Proceedings of the 4th ACM Conference on Recommender Systems*. RecSys ’10. 2010, pp. 257–260 (cit. on p. 10).

[GH16] Carlos A. Gomez-Uribe and Neil Hunt. “The Netflix Recommender System: Algorithms, Business Value, and Innovation”. In: *ACM Transactions on Management Information Systems* 6.4 (Jan. 2016), 13:1–13:19 (cit. on pp. 1, 54).

[Grb+15] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, et al. “E-commerce in Your Inbox: Product Recommendations at Scale”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’15. 2015, pp. 1809–1818 (cit. on pp. 10, 36).

[Guo+19] Huifeng Guo, Ruiming Tang, Yunming Ye, Feng Liu, and Yuzhou Zhang. “A Novel KNN Approach for Session-Based Recommendation”. In: *Proceedings of the 23rd Pacific-Asia Conference on Knowledge Discovery and Data Mining*. PAKDD ’19. 2019, pp. 381–393 (cit. on p. 35).

[GWD14] Alex Graves, Greg Wayne, and Ivo Danihelka. “Neural Turing Machines”. In: *CoRR* abs/1410.5401 (2014). arXiv: 1410.5401 (cit. on p. 35).

[He+09] Qi He, Daxin Jiang, Zhen Liao, Steven C. H. Hoi, Kuiyu Chang, Ee-Peng Lim, et al. “Web Query Recommendation via Sequential Query Prediction”. In: *Proceedings of the 2009 IEEE International Conference on Data Engineering*. ICDE ’09. 2009, pp. 1443–1454 (cit. on p. 36).

[He+15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR ’15. 2015, pp. 770–778 (cit. on p. 34).

[He+16] Jing He, Xin Li, Lejian Liao, Dandan Song, and William Cheung. “Inferring a Personalized Next Point-of-Interest Recommendation Model with Latent Behavior Patterns”. In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. AAAI ’16. 2016, pp. 137–143 (cit. on pp. 27–29).

[Her+04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. “Evaluating Collaborative Filtering Recommender Systems”. In: *ACM Transactions on Information Systems* 22.1 (Jan. 2004), pp. 5–53 (cit. on p. 10).

[Hid+16a] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. “Session-based Recommendations with Recurrent Neural Networks”. In: *Proceedings of the 4th International Conference on Learning Representations*. ICLR ’16. 2016 (cit. on pp. 7, 9, 21, 24, 30, 31, 37–39, 46).

[Hid+16b] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. “Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations”. In: *Proceedings of the 10th ACM Conference on Recommender Systems*. RecSys ’16. 2016, pp. 241–248 (cit. on pp. 7, 35).

[HK18] Balázs Hidasi and Alexandros Karatzoglou. “Recurrent Neural Networks with Top-k Gains for Session-based Recommendations”. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. CIKM ’18. 2018, pp. 843–852 (cit. on pp. 30, 31, 35).

[HKJ03] Taher Haveliwala, Sepandar Kamvar, and Glen Jeh. *An Analytical Comparison of Approaches to Personalizing PageRank*. Technical Report 2003-35. Stanford InfoLab, June 2003 (cit. on p. 67).

[HKV08] Yifan Hu, Yehuda Koren, and Chris Volinsky. “Collaborative Filtering for Implicit Feedback Datasets”. In: *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*. ICDM ’08. 2008, pp. 263–272 (cit. on p. 4).

[HLC08] Sue-Chen Hsueh, Ming-Yen Lin, and Chien-Liang Chen. “Mining Negative Sequential Patterns for E-commerce Recommendations”. In: *Proceedings of the 3rd IEEE Asia-Pacific Services Computing Conference*. APSCC ’08. 2008, pp. 1213–1218 (cit. on p. 7).

[HMB12] Negar Hariri, Bamshad Mobasher, and Robin Burke. “Context-aware Music Recommendation Based on Latenttopic Sequential Patterns”. In: *Proceedings of the 6th ACM Conference on Recommender Systems*. RecSys ’12. 2012, pp. 131–138 (cit. on pp. 7, 9, 21, 34).

[Hos+15] Mehdi Hosseinzadeh Aghdam, Negar Hariri, Bamshad Mobasher, and Robin Burke. “Adapting Recommendations to Contextual Changes Using Hierarchical Hidden Markov Models”. In: *Proceedings of the 9th ACM Conference on Recommender Systems*. RecSys ’15. 2015, pp. 241–244 (cit. on p. 36).

[JA16] Dietmar Jannach and Gediminas Adomavicius. “Recommendations with a Purpose”. In: *Proceedings of the 10th ACM Conference on Recommender Systems*. RecSys ’16. 2016, pp. 7–10 (cit. on p. 37).

[JH09] Dietmar Jannach and Kolja Hegelich. “A Case Study on the Effectiveness of Recommendations in the Mobile Internet”. In: *Proceedings of the 3rd ACM Conference on Recommender Systems*. RecSys ’09. 2009, pp. 205–208 (cit. on p. 4).

[JJ19] Dietmar Jannach and Michael Jugovac. “Measuring the Business Value of Recommender Systems”. In: *ACM Transactions on Management Information Systems* 10.4 (2019) (cit. on pp. 1, 10).

[JKG12] Dietmar Jannach, Zeynep Karakaya, and Fatih Gedikli. “Accuracy Improvements for Multi-criteria Recommender Systems”. In: *Proceedings of the 13th ACM Conference on Electronic Commerce*. EC ’12. 2012, pp. 674–689 (cit. on p. 4).

[JL17a] Dietmar Jannach and Malte Ludewig. “Determining Characteristics of Successful Recommendations from Log Data: A Case Study”. In: *Proceedings of the 32nd ACM SIGAPP Symposium On Applied Computing*. SAC ’17. 2017, pp. 1643–1648 (cit. on pp. 14, 15, 53–55, 60, 67, 89).

[JL17b] Dietmar Jannach and Malte Ludewig. “Investigating Personalized Search in E-Commerce”. In: *Proceedings of the 30th International Florida Artificial Intelligence Research Society Conference*. FLAIRS ’17. 2017, pp. 645–650 (cit. on pp. 14, 15, 53, 65, 67, 68, 89).

[JL17c] Dietmar Jannach and Malte Ludewig. “When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation”. In: *Proceedings of the 11th ACM Conference on Recommender Systems*. RecSys ’17. 2017, pp. 306–310 (cit. on pp. 13, 15, 16, 20, 24, 25, 30, 35, 39, 40, 45, 46, 89).

[JLJ15a] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. “Adaptation and Evaluation of Recommendations for Short-term Shopping Goals”. In: *Proceedings of the 9th ACM Conference on Recommender Systems*. RecSys ’15. 2015, pp. 211–218 (cit. on pp. 7, 9, 11, 53, 55, 59, 60, 62, 63).

[JLJ15b] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. “Item Familiarity as a Possible Confounding Factor in User-Centric Recommender Systems Evaluation”. In: *i-com Journal of Interactive Media* 14.1 (2015), pp. 29–39 (cit. on p. 47).

[JLL17] Dietmar Jannach, Malte Ludewig, and Lukas Lerche. “Session-Based Item Recommendation in E-Commerce: On Short-Term Intents, Reminders, Trends and Discounts”. In: *User Modeling and User-Adapted Interaction* 27.3-5 (2017), pp. 351–392 (cit. on pp. 14, 15, 45, 53, 55, 57, 58, 60, 62–64, 67, 89).

[JLZ18] Dietmar Jannach, Lukas Lerche, and Markus Zanker. “Recommending Based on Implicit Feedback”. In: *Social Information Access: Systems and Technologies*. Ed. by Peter Brusilovsky and Daqing He. Cham: Springer International Publishing, 2018, pp. 510–569 (cit. on p. 4).

[JWK14] Gawesh Jawaheer, Peter Weller, and Patty Kostkova. “Modeling User Preferences in Recommender Systems: A Classification Framework for Explicit and Implicit User Feedback”. In: *ACM Transactions on Interactive Intelligent Systems* 4.2 (June 2014), 8:1–8:26 (cit. on p. 3).

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. “Matrix Factorization Techniques for Recommender Systems”. In: *Computer* 42.8 (Aug. 2009), pp. 30–37 (cit. on p. 2).

[KJ17] Iman Kamehkhosh and Dietmar Jannach. “User Perception of Next-Track Music Recommendations”. In: *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. UMAP ’17. 2017, pp. 113–121 (cit. on pp. 10, 12, 47).

[KJL17] Iman Kamehkhosh, Dietmar Jannach, and Malte Ludewig. “A Comparison of Frequent Pattern Techniques and a Deep Learning Method for Session-Based Recommendation”. In: *Proceedings of the 1st Workshop on Temporal Reasoning in Recommender Systems co-located with 11th ACM Conference on Recommender Systems*. RecTemp ’17. 2017, pp. 50–56 (cit. on pp. 13, 16, 20, 23, 40, 89).

[Kni+12] Bart P. Knijnenburg, Martijn C. Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. “Explaining the User Experience of Recommender Systems”. In: *User Modeling and User-Adapted Interaction* 22.4 (Oct. 2012), pp. 441–504 (cit. on p. 49).

[KNK13] Santosh Kabbur, Xia Ning, and George Karypis. “FISM: Factored Item Similarity Models for top-N Recommender Systems”. In: *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’13. 2013, pp. 659–667 (cit. on pp. 27, 28).

[Kor09] Yehuda Koren. “The BellKor Solution to the Netflix Grand Prize”. In: *Netflix Prize Documentation* 81.2009 (2009), pp. 1–10 (cit. on pp. 2, 3).

[Li+17] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. “Neural Attentive Session-based Recommendation”. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. CIKM ’17. 2017, pp. 1419–1428 (cit. on pp. 21, 31, 32, 37, 39).

[Liu+14] Zitao Liu, Gyanit Singh, Nish Parikh, and Neel Sundaresan. “A Large Scale Query Logs Analysis for Assessing Personalization Opportunities in E-commerce Sites”. In: *Proceedings of the WSCD Workshop at the 7th ACM International Conference on Web Search and Data Mining*. WSCD ’14. 2014 (cit. on p. 65).

[Liu+18] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. “STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’18. 2018, pp. 1831–1839 (cit. on pp. 21, 30, 32, 33, 37, 39).

[LJ18a] Malte Ludewig and Dietmar Jannach. “Could You Play That Song Again? - Reminding Users of Their Favorite Tracks Through Recommendations”. In: *Proceedings of the WSDM Cup Workshop 2018*. WSDM Cup ’18. 2018 (cit. on pp. 18, 90).

[LJ18b] Malte Ludewig and Dietmar Jannach. “Evaluation of Session-Based Recommendation Algorithms”. In: *User Modeling and User-Adapted Interaction* 28.4-5 (2018), pp. 331–390 (cit. on pp. 13, 16, 17, 20, 24, 25, 28–30, 36–40, 42, 45, 62, 89).

[LJ19a] Malte Ludewig and Dietmar Jannach. “Learning to Rank Hotels for Search and Recommendation from Session-Based Interaction Logs and Meta Data”. In: *Proceedings of the ACM Recommender Systems Challenge 2019*. RecSys Challenge ’19. 2019 (cit. on pp. 18, 90).

[LJ19b] Malte Ludewig and Dietmar Jannach. “User-Centric Evaluation of Session-Based Recommendations for an Automated Radio Station”. In: *Proceedings of the 13th ACM Conference on Recommender Systems*. RecSys ’19. 2019, pp. 306–310 (cit. on pp. 14, 17, 20, 40, 47, 49, 51, 89).

[LJJ17] Malte Ludewig, Michael Jugovac, and Dietmar Jannach. “A Light-Weight Approach to Recipient Determination When Recommending New Items”. In: *Proceedings of the ACM Recommender Systems Challenge 2017*. RecSys Challenge ’17. 2017, 3:1–3:6 (cit. on pp. 16, 17, 90).

[LJL16] Lukas Lerche, Dietmar Jannach, and Malte Ludewig. “On the Value of Reminders within E-Commerce Recommendations”. In: *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*. UMAP ’16. 2016, pp. 27–35 (cit. on pp. 15, 55, 61, 90).

[LLC14] Eric Hsueh-Chan Lu, Yi-Wei Lin, and Jing-Bin Ciou. “Mining mobile application sequential patterns for usage prediction”. In: *Proceedings of the 2014 IEEE International Conference on Granular Computing*. GrC ’14. Oct. 2014, pp. 185–190 (cit. on p. 7).

[LLH17] Pablo Loyola, Chen Liu, and Yu Hirate. “Modeling User Session and Intent with an Attention-based Encoder-Decoder Architecture”. In: *Proceedings of the 11th ACM Conference on Recommender Systems*. RecSys ’17. 2017, pp. 147–151 (cit. on p. 35).

[Loe+18] Benedikt Loepp, Tim Donkers, Timm Kleemann, and Jürgen Ziegler. “Impact of Item Consumption on Assessment of Recommendations in User Studies”. In: *Proceedings of the 12th ACM Conference on Recommender Systems*. RecSys ’18. 2018, pp. 49–53 (cit. on p. 47).

[LSY03] Greg Linden, Brent Smith, and Jeremy York. “Amazon.com recommendations: item-to-item collaborative filtering”. In: *IEEE Internet Computing* 7.1 (Jan. 2003), pp. 76–80 (cit. on p. 24).

[Lud+18] Malte Ludewig, Iman Kamehkhosh, Nick Landia, and Dietmar Jannach. “Effective Nearest-Neighbor Music Recommendations”. In: *Proceedings of the ACM Recommender Systems Challenge 2018*. RecSys Challenge ’18. 2018, 3:1–3:6 (cit. on pp. 18, 26, 49, 90).

[Lud+19] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. “Performance Comparison of Neural and Non-Neural Approaches to Session-Based Recommendation”. In: *Proceedings of the 13th ACM Conference on Recommender Systems*. RecSys ’19. 2019, pp. 306–310 (cit. on pp. 13, 16, 20, 24, 26, 30, 38–40, 90).

[MBR12] Omar Moling, Linas Baltrunas, and Francesco Ricci. “Optimal Radio Channel Recommendations with Explicit and Implicit Feedback”. In: *Proceedings of the Sixth ACM Conference on Recommender Systems*. RecSys ’12. 2012, pp. 75–82 (cit. on pp. 10, 36).

[MF18] Fei Mi and Boi Faltings. *Context Tree for Adaptive Session-based Recommendation*. 2018. arXiv: `1806.03733 [cs.IR]` (cit. on pp. 22, 23).

[Mik+13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS ’13. 2013, pp. 3111–3119 (cit. on p. 36).

[MKP03] Harry Mak, Irena Koprinska, and Josiah Poon. “INTIMATE: A Web-Based Movie Recommender Using Text Categorization”. In: *Proceedings of the IEEE/WIC International Conference on Web Intelligence*. WI ’03. Oct. 2003, pp. 602–605 (cit. on p. 2).

[ML11] Brian McFee and Gert Lanckriet. “The Natural Language of Playlists”. In: *Proceedings of the 12th International Society for Music Information Retrieval Conference*. ISMIR ’11. Jan. 2011, pp. 537–542 (cit. on p. 36).

[ML12] Brian Mcfee and Gert Lanckriet. “Hypergraph Models of Playlist Dialects”. In: *Proceedings of the 13th International Society for Music Information Retrieval Conference*. ISMIR ’12. Oct. 2012 (cit. on p. 39).

[Mob+02] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. “Using Sequential and Non-Sequential Patterns in Predictive Web Usage Mining Tasks”. In: *Proceedings of the 2002 IEEE International Conference on Data Mining*. ICDM ’02. Dec. 2002, pp. 669–672 (cit. on pp. 21, 34).

[MR11] Nicolaas Matthijs and Filip Radlinski. “Personalizing Web Search Using Long Term Browsing History”. In: *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*. WSDM ’11. 2011, pp. 25–34 (cit. on pp. 67, 68).

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008 (cit. on p. 58).

[Oor+16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, et al. *WaveNet: A Generative Model for Raw Audio*. 2016. arXiv: `1609.03499 [cs.SD]` (cit. on pp. 33, 34).

[PCH11] Pearl Pu, Li Chen, and Rong Hu. “A User-centric Evaluation Framework for Recommender Systems”. In: *Proceedings of the Fifth ACM Conference on Recommender Systems*. RecSys ’11. 2011, pp. 157–164 (cit. on p. 49).

[Pla+06] Carolin Plate, Nathalie Basselin, Alexander Kröner, Michael Schneider, Stephan Baldes, Vania Dimitrova, et al. “Recomindation: New Functions for Augmented Memories”. In: *Proceedings of the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*. AH ’06. June 2006, pp. 141–150 (cit. on p. 56).

[PS11] Nish Parikh and Neel Sundaresan. “A user-tunable approach to marketplace search”. In: *Proceedings of the 20th International Conference on World Wide Web*. WWW ’11. 2011, pp. 245–248 (cit. on p. 65).

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. “Glove: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*. EMNLP ’14. 2014, pp. 1532–1543 (cit. on p. 36).

[QCJ18] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. “Sequence-Aware Recommender Systems”. In: *ACM Computing Surveys* 51.4 (Sept. 2018), 66:1–66:36 (cit. on pp. 6–9, 19, 36).

[Ren+09] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. “BPR: Bayesian Personalized Ranking from Implicit Feedback”. In: *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. UAI ’09. 2009, pp. 452–461 (cit. on pp. 4, 27, 36).

[Ren+19] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. “RepeatNet: A Repeat Aware Neural Recommendation Machine for Session-Based Recommendation”. In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. AAAI ’19. 2019, pp. 4806–4813 (cit. on p. 35).

[RFS10] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. “Factorizing Personalized Markov Chains for Next-basket Recommendation”. In: *Proceedings of the 19th International Conference on World Wide Web*. WWW ’10. 2010, pp. 811–820 (cit. on pp. 21, 27, 28).

[RLJ16] Siddharth Reddy, Igor Labutov, and Thorsten Joachims. “Learning Student and Content Embeddings for Personalized Lesson Sequence Recommendation”. In: *Proceedings of the 3rd ACM Conference on Learning @ Scale*. L@S ’16. 2016, pp. 93–96 (cit. on p. 36).

[Sar+01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. “Item-based Collaborative Filtering Recommendation Algorithms”. In: *Proceedings of the 10th International Conference on World Wide Web*. WWW ’01. 2001, pp. 285–295 (cit. on p. 3).

[Sca+09] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. “The Graph Neural Network Model”. In: *IEEE Transactions on Neural Networks* 20.1 (Jan. 2009), pp. 61–80 (cit. on p. 35).

[Sch+16] Tobias Schnabel, Paul N. Bennett, Susan T. Dumais, and Thorsten Joachims. “Using Shortlists to Support Decision Making and Improve Recommender System Performance”. In: *Proceedings of the 25th International Conference on World Wide Web*. WWW ’16. 2016, pp. 987–997 (cit. on pp. 53, 56).

[SEH16] Yang Song, Ali Mamdouh Elkahky, and Xiaodong He. “Multi-Rate Deep Learning for Temporal Recommendation”. In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’16. 2016, pp. 909–912 (cit. on pp. 7, 36).

[SFC18] Gabriel de Souza Pereira Moreira, Felipe Ferreira, and Adilson Marques da Cunha. “News Session-Based Recommendations Using Deep Neural Networks”. In: *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems*. DLRS ’18. 2018, pp. 15–23 (cit. on p. 35).

[SHB05] Guy Shani, David Heckerman, and Ronen I. Brafman. “An MDP-Based Recommender System”. In: *The Journal of Machine Learning Research* 6 (Dec. 2005), pp. 1265–1295 (cit. on pp. 36, 53).

[SMB07] Ahu Sieg, Bamshad Mobasher, and Robin Burke. “Web Search Personalization with Ontological User Profiles”. In: *Proceedings of the 16th ACM Conference on Conference on Information and Knowledge Management*. CIKM ’07. 2007, pp. 525–534 (cit. on p. 67).

[Soh+17] Harold Soh, Scott Sanner, Madeleine White, and Greg Jamieson. “Deep Sequential Recommendation for Personalized Adaptive User Interfaces”. In: *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. IUI ’17. 2017, pp. 589–593 (cit. on p. 36).

[Sor+15] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. “A Hierarchical Recurrent Encoder-Decoder for Generative Context-Aware Query Suggestion”. In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM ’15. 2015, pp. 553–562 (cit. on p. 36).

[SSZ18] Gabriele Sottocornola, Panagiotis Symeonidis, and Markus Zanker. “Session-based News Recommendations”. In: *Companion Proceedings of the The Web Conference 2018*. WWW ’18 Companion. 2018, pp. 1395–1399 (cit. on p. 35).

[Tag+15] Yukihiro Tagami, Hayato Kobayashi, Shingo Ono, and Akira Tajima. “Modeling User Activities on the Web Using Paragraph Vector”. In: *Companion Proceedings of the 24th International Conference on World Wide Web*. WWW ’15 Companion. 2015, pp. 125–126 (cit. on p. 36).

[TB14] Maryam Tavakol and Ulf Brefeld. “Factored MDPs for Detecting Topics of User Sessions”. In: *Proceedings of the 8th ACM Conference on Recommender Systems*. RecSys ’14. 2014, pp. 33–40 (cit. on pp. 36, 53).

[TPT11] Gábor Takács, István Pilászy, and Domonkos Tikk. “Applications of the Conjugate Gradient Method for Implicit Feedback Collaborative Filtering”. In: *Proceedings of the 5th ACM Conference on Recommender Systems*. RecSys ’11. 2011, pp. 297–300 (cit. on p. 4).

[TSZ06] Bin Tan, Xuehua Shen, and ChengXiang Zhai. “Mining Long-term Search History to Improve Search Accuracy”. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’06. 2006, pp. 718–723 (cit. on p. 67).

[Tur+15] Roberto Turrin, Massimo Quadrana, Andrea Condorelli, Roberto Pagano, and Paolo Cremonesi. “30Music Listening and Playlists Dataset”. In: *Poster Proceedings of the 9th ACM Conference on Recommender Systems*. RecSys ’15. 2015 (cit. on p. 39).

[Twa16] Bartlomiej Twardowski. “Modelling Contextual Information in Session-Aware Recommender Systems with Neural Networks”. In: *Proceedings of the 10th ACM Conference on Recommender Systems*. RecSys ’16. 2016, pp. 273–276 (cit. on pp. 7, 36).

[TXL16] Yong Kiam Tan, Xinxing Xu, and Yong Liu. “Improved Recurrent Neural Networks for Session-based Recommendations”. In: *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. DLRS ’16. 2016, pp. 17–22 (cit. on p. 35).

[VG14] Koen Verstrepen and Bart Goethals. “Unifying Nearest Neighbors Collaborative Filtering”. In: *Proceedings of the 8th ACM Conference on Recommender Systems*. RecSys ’14. 2014, pp. 177–184 (cit. on p. 24).

[VSC16] Flavian Vasile, Elena Smirnova, and Alexis Conneau. “Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation”. In: *Proceedings of the 10th ACM Conference on Recommender Systems*. RecSys ’16. 2016, pp. 225–232 (cit. on p. 36).

[Wan+19] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten de Rijke. “A Collaborative Session-based Recommendation Approach with Parallel Memory Modules”. In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR’19. 2019, pp. 345–354 (cit. on pp. 30, 34, 35).

[WCB15] Jason Weston, Sumit Chopra, and Antoine Bordes. “Memory Networks”. In: *Proceedings of the 3rd International Conference on Learning Representations*. ICLR ’15. 2015 (cit. on p. 35).

[Wu+13] Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, et al. “Personalized Next-song Recommendation in Online Karaokes”. In: *Proceedings of the 7th ACM Conference on Recommender Systems*. RecSys ’13. 2013, pp. 137–140 (cit. on p. 36).

[Wu+19] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. “Session-based Recommendation with Graph Neural Networks”. In: *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*. AAAI ’19. July 2019, pp. 346–353 (cit. on pp. 30, 34, 35).

[YLY12] Ghim-Eng Yap, Xiao-Li Li, and Philip S. Yu. “Effective Next-items Recommendation via Personalized Sequential Pattern Mining”. In: *Proceedings of the 17th International Conference on Database Systems for Advanced Applications*. DASFAA ’12. 2012, pp. 48–64 (cit. on pp. 21, 34).

[Yu+14] Jun Yu, Sunil Mohan, Duangmanee Putthividhya, and Weng-Keen Wong. “Latent Dirichlet Allocation Based Diversified Retrieval for e-Commerce Search”. In: *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. WSDM ’14. 2014, pp. 463–472 (cit. on p. 65).

[Yu+16] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. “A Dynamic Recurrent Model for Next Basket Recommendation”. In: *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’16. 2016, pp. 729–732 (cit. on p. 36).

[Yua+19] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. “A Simple Convolutional Generative Network for Next Item Recommendation”. In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. WSDM ’19. 2019, pp. 582–590 (cit. on pp. 30, 33, 34).

[Zan+14] Eva Zangerle, Martin Pichl, Wolfgang Gassler, and Günther Specht. “#nowplaying Music Dataset: Extracting Listening Behavior from Twitter”. In: *Proceedings of the First International Workshop on Internet-Scale Multimedia Management*. WISMM ’14. 2014, pp. 21–26 (cit. on p. 39).

[Zha+14] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, et al. “Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks”. In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. AAAI ’14. 2014, pp. 1369–1375 (cit. on p. 36).

[Zhe+10] Elena Zheleva, John Guiver, Eduarda Mendes Rodrigues, and Nataša Milić-Frayling. “Statistical Models of Music-listening Sessions in Social Media”. In: *Proceedings of the 19th International Conference on World Wide Web*. WWW ’10. 2010, pp. 1019–1028 (cit. on p. 36).

[ZLW12a] Dong Zhou, Séamus Lawless, and Vincent Wade. “Improving Search via Personalized Query Expansion Using Social Media”. In: *Information Retrieval* 15.3-4 (June 2012), pp. 218–242 (cit. on p. 67).

[ZLW12b] Dong Zhou, Séamus Lawless, and Vincent Wade. “Web Search Personalization Using Social Data”. In: *Proceedings of Theory and Practice of Digital Libraries - Second International Conference*. TPDL ’12. 2012, pp. 298–310 (cit. on p. 67).

Web pages

[Bre15] Bree Brouwer. *YouTube Now Gets Over 400 Hours Of Content Uploaded Every Minute*. 2015. URL: `https://www.tubefilter.com/2015/07/26/youtube-400-hours-content-every-minute` (visited on Dec. 30, 2019) (cit. on p. 1).

[Fun07] Simon Funk. *Netflix update: Try this at home.* 2007. URL: `http://sifter.org/~simon/journal/20061211.html` (visited on Dec. 30, 2019) (cit. on p. 3).

[Spo19] Spotify AB. *Company Info*. 2019. URL: `https://newsroom.spotify.com/company-info` (visited on Dec. 30, 2019) (cit. on p. 1).

List of Figures

List of Tables

Publications

In this thesis the following eight works of the author are covered. The full texts of these works can be found after this list.

- Dietmar Jannach and Malte Ludewig. “Determining Characteristics of Successful Recommendations from Log Data: A Case Study”. In: *Proceedings of the 32nd ACM SIGAPP Symposium On Applied Computing*. SAC ’17. 2017, pp. 1643–1648.
- Dietmar Jannach and Malte Ludewig. “Investigating Personalized Search in E-Commerce”. In: *Proceedings of the 30th International Florida Artificial Intelligence Research Society Conference*. FLAIRS ’17. 2017, pp. 645–650.
- Dietmar Jannach, Malte Ludewig, and Lukas Lerche. “Session-Based Item Recommendation in E-Commerce: On Short-Term Intents, Reminders, Trends and Discounts”. In: *User Modeling and User-Adapted Interaction* 27.3-5 (2017), pp. 351–392.
- Dietmar Jannach and Malte Ludewig. “When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation”. In: *Proceedings of the 11th ACM Conference on Recommender Systems*. RecSys ’17. 2017, pp. 306–310.
- Iman Kamehkhosh, Dietmar Jannach, and Malte Ludewig. “A Comparison of Frequent Pattern Techniques and a Deep Learning Method for Session-Based Recommendation”. In: *Proceedings of the 1st Workshop on Temporal Reasoning in Recommender Systems co-located with 11th ACM Conference on Recommender Systems*. RecTemp ’17. 2017, pp. 50–56.
- Malte Ludewig and Dietmar Jannach. “Evaluation of Session-Based Recommendation Algorithms”. In: *User Modeling and User-Adapted Interaction* 28.4-5 (2018), pp. 331–390.
- Malte Ludewig and Dietmar Jannach. “User-Centric Evaluation of Session-Based Recommendations for an Automated Radio Station”. In: *Proceedings of the 13th ACM Conference on Recommender Systems*. RecSys ’19. 2019, pp. 306–310.

- Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. "Performance Comparison of Neural and Non-Neural Approaches to Session-Based Recommendation". In: *Proceedings of the 13th ACM Conference on Recommender Systems*. RecSys '19. 2019, pp. 306–310.

In addition to these eight main publications, the author of this thesis worked on the following other publications related to recommender systems that are not included in this thesis.

- Lukas Lerche, Dietmar Jannach, and Malte Ludewig. "On the Value of Reminders within E-Commerce Recommendations". In: *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*. UMAP '16. 2016, pp. 27–35.
- Malte Ludewig, Michael Jugovac, and Dietmar Jannach. "A Light-Weight Approach to Recipient Determination When Recommending New Items". In: *Proceedings of the ACM Recommender Systems Challenge 2017*. RecSys Challenge '17. 2017, 3:1–3:6.
- Malte Ludewig and Dietmar Jannach. "Could You Play That Song Again? - Reminding Users of Their Favorite Tracks Through Recommendations". In: *Proceedings of the WSDM Cup Workshop 2018*. WSDM Cup '18. 2018.
- Malte Ludewig, Iman Kamehkhosh, Nick Landia, and Dietmar Jannach. "Effective Nearest-Neighbor Music Recommendations". In: *Proceedings of the ACM Recommender Systems Challenge 2018*. RecSys Challenge '18. 2018, 3:1–3:6.
- Malte Ludewig and Dietmar Jannach. "Learning to Rank Hotels for Search and Recommendation from Session-Based Interaction Logs and Meta Data". In: *Proceedings of the ACM Recommender Systems Challenge 2019*. RecSys Challenge '19. 2019.

Determining Characteristics of Successful Recommendations from Log Data – A Case Study

Dietmar Jannach
TU Dortmund, Germany
dietmar.jannach@tu-dortmund.de

Malte Ludewig
TU Dortmund, Germany
malte.ludewig@tu-dortmund.de

ABSTRACT

Academic research in recommender systems largely focuses on the problem of predicting the relevance of (long-tail) items that the individual user presumably does not know yet. Many real-world systems however also recommend items that users have inspected in the past, items that are popular at the moment, and items currently on sale. In this work we investigate the value of including such items in recommendation lists based on an analysis of the web logs of a large online retailer. An examination of the features of successful item suggestions reveals that the chances of a recommendation leading to a purchase increase when the item is recently trending, on sale, or was recently viewed by the user. Offline simulation experiments furthermore show that considering those success factors that were identified from log data in the ranking algorithms can help to increase the prediction accuracy of recommender systems.

CCS Concepts

•**Information systems** → **Recommender systems;**
•**Applied computing** → **Electronic commerce;**

Keywords

Recommender Systems; Success Factors; Case Study

1. INTRODUCTION

Automated recommendations of the type "You may also be interested in" are common on today's e-commerce sites, and ample evidence exists that such personalized or item-related recommendations can measurably impact businesses [2, 4, 5, 6, 9, 10]. Recommender systems are correspondingly an active area of research, and significant advances have been made in recent years in terms of accurately predicting the relevance of individual items for each user.

The most common research approach in the field is to consider only items for recommendation that the user has not seen, consumed, or purchased before. In addition, recommending items which are unknown to the user, but very popular in general, is often considered to be of limited value for users, and sometimes such items are explicitly excluded from the evaluation as done, for example, in [13].

Real-world e-commerce sites like Amazon.com, however, include items in their recommendation lists that the users have just recently inspected and are therefore at least aware of or already familiar with. Furthermore, recommending popular items like top sellers or recently trending items is quite common in this domain. Finally, an additional aspect of real-world online shops barely investigated in the literature is that some recommended items can be on sale and correspondingly labeled with discount tags.

In a previous work [10], the authors analyzed the first of these aspects by examining the value of including *reminders*, i.e., items that the user has inspected in the past, into recommendation lists. For these analyses, the authors relied on specific log data of a large online retailer, which contains information about (a) which items were recommended to customers and (b) which items were actually purchased from the recommendation list. The analyses showed that many of the recommendations that were clicked on and later on purchased were already known to the users.

In this work we investigate – using the same log data – if there are other seldom-researched factors in e-commerce settings that contribute to the success of a recommendation. To identify such factors, we conduct a broader systematic analysis of the given user interaction data. Based on the results, we will particularly focus on the effectiveness of including *trending items* and *items on sale* into recommendations. Both types of recommendations are not uncommon in practice as mentioned above. We additionally propose first algorithms that take these aspects into account in the item ranking process and test if the inclusion of this additional information helps to increase the ranking accuracy of recommendation algorithms.

Obviously, recommending *only* discounted or currently trending items might in most cases be less effective in terms of the business value. This was shown, e.g., in [6], where the recommendation of top sellers was not very effective in terms of sales. However, we show that including *some* known or trending items can be valuable and also that many users actually inspect the recommended on-sale items. Overall, our work therefore contributes to a better understanding of what makes a recommendation successful in practice, at least in the domain of fashion products that we comprehensively analyze in this paper.

SAC 2017, April 03 - 07, 2017, Marrakech, Morocco

ISBN 978-1-4503-4486-9/17/04…$15.00

DOI: http://dx.doi.org/10.1145/3019612.3019757

2. DATA ANALYSIS

We base our analyses on a dataset of user navigation logs of the large European fashion retailer Zalando. In the following, we first introduce characteristics of the dataset itself and the subsets we use. Then, we investigate properties of successful recommendations with the help of basic statistical analyses. Finally, we present the results of a systematic feature analysis.

2.1 Dataset Properties

The raw dataset contains website interaction events of about 3.5 million users. As in [8], the logged actions include typical events such as item views, purchases, and add-to-cart actions. Each action relates to one of over 400,000 catalog items. For each item basic information is available to us, e.g., the brand, the category, or if the item was on sale at that time. The dataset is anonymized and distorted in a way that no inference about business figures of the shop is possible.

An additional unique feature of our dataset is that it contains (a) information about *which recommendations were made* to users when they inspected the details of an item[1] and (b) click events on these recommended items. This information allows us to investigate *which features make recommendations successful in this domain.*

The raw dataset is very sparse and contains many users who have only visited the shop once or never made a purchase. We therefore created two data samples as in [8] and [10]. One consists of 3,000 *heavy* users who visited the shop frequently. The other one consists of 3,000 *occasional* users that were randomly selected from a subset of users for which we recorded at least 10 purchase events. Over the period of one year, *heavy* users visited the shop about two times a week, resulting in about 114 sessions per year. *Occasional* users on average interacted with the shop every second week (about 28 sessions).

2.2 Properties of Successful Recommendations

General Conversion Rate. We use *conversion rates* as the usual *success* measure for our analysis. Using the dataset of *heavy* users as a basis, we observe that about every 100th recommendation list attracted a click. Note that the absolute number depends on several factors, e.g., on the visual layout of the shop, and cannot be compared across shops. Our analysis in addition shows that in as many as 14% of the cases when a recommended item was actually clicked-on (viewed), it was added to the shopping cart in the current or next two sessions. Furthermore, about 7% of the recommended-item clicks actually led to a purchase of the item. This indicates that while recommendation lists are generally not a main navigation mechanism for users on the site, they lead to high "click-to-buy" conversions once the users have found a potentially relevant item within the recommendations.

Recommendation of Familiar and New Items. Continuing the research in [10], we examined to what extent recommending items that users already know can be a successful strategy. From all recommended items, about 10% were goods that the users had inspected on the site before. This percentage depends on the specific inner workings of the used algorithms on the site, which are unknown to us. It is therefore more interesting that nearly half of the *successful* recommendations (44%) were not new to the users, which highlights the potential value of including reminders in recommendation lists [10].

On the other hand, this also means that more than half of the recommendation-induced purchases were related to items that the consumers have not seen before. This can be seen as additional evidence of the capabilities of recommenders to help users discover relevant items. The main practical implication of the combined findings is however that including at least some already known (but not yet purchased) items can represent a promising strategy to increase the overall effectiveness of the recommender system.

The Importance of Short-Term Shopping Intents. The investigations in [8] showed – using simulation experiments on real data – that adapting to the user's immediate shopping intent is important to increase the prediction accuracy of recommendation algorithms[2]. To understand to what extent online consumers are focused on a goal when they visit the site, we analyzed their browsing behavior. On average, users inspected about 9 different items from 2.7 (of more than 330 available) categories, and considered 2.5 different colors and 3.6 different brands during one session. These numbers suggest that users indeed often have a specific shopping goal in each session. Thus, recommending items that are similar to the estimated shopping intent seems promising.

To quantify this aspect, we compared the recommend-to-purchase conversion rate when items were recommended that were similar to the most recent item to the conversion rate when this was not the case. The results are shown in Table 1 and we can see that recommending items that, e.g., have the same color as the currently viewed one, leads to a substantial increase of the conversion rate.

Table 1: Recommend-to-purchase conversions for similar-item and different-item recommendations

Item feature	Different value	Same value	Difference
Brand	0.950 %	4.227 %	345 %
Price level	1.403 %	3.624 %	158 %
Category	1.207 %	2.844 %	135 %
Color	1.521 %	2.701 %	77 %

The difference in the conversion rates lie between about 77% (color attribute) and more than 300% for the brand. The dominating importance of the brand is not particularly surprising in the fashion domain. However, consumers also seem to have a strong preference for certain price ranges (which were predefined categories in our dataset). Also, recommending items from the same sub-category and with the same color led to high increases of the conversion rates.

In sum, recommendations were more successful when the recommended items were very similar to the currently inspected ones. This is consistent with the observations regarding the focused shopping behavior of many users discussed above. Also, this underlines that in the examined domain the recommendation of substitute products is clearly

[1]Sets of three recommendations were recorded.

[2]A similar approach has shown to work well for news recommendations [11].

advantageous when compared to the recommendation of alternatives, which corroborates some observations from [3]. The accuracy results from the experiments in [8], which included a recommendation component that considers the features of recently viewed items, also confirm this hypothesis.

Considering Popular and Trending Items. Recommending popular items is generally a "safe" strategy, even though it might not lead to the highest business value as shown, e.g., in [6]. The fact that not all items are equally popular in a shop was also reflected in the consumer's adoption of the recommendations. When one of the three recommendations of a list was inspected by a user, the chances that it was the most popular one among the three – measured in terms of clicks and purchases by the entire user community – were at 43%, i.e., measurably higher than the theoretical 33% random chance.

Since seasonal trends are common in the fashion domain, we furthermore looked at the most recent popularity of the items. When looking for example at the popularity of the items on the day of the recommendations, we found that recommendations were particularly successful when they concerned these recently trending items. Using a normalized popularity score, the daily average popularity of all recommended items was at 0.024, whereas the average of those which were actually selected afterwards was at 0.088, i.e., three times higher.[3]

The Role of Discounts. The pricing of items usually has a direct impact on demand levels and sales. It therefore stands to reason that items in recommendation lists which are marked as being on sale and which are discounted are more attractive for users than other items. On the other hand, recommendations that contain too many or only discounted items, might raise the impression that the item suggestions are biased and the presented list is rather an advertisement than a recommendation.

In our dataset, we know for each recommended item if it was on discount at the time of the recommendation or not. We could therefore compare the recommendation-to-buy conversion rates for on-sale items and regular-prized items. The observed differences were huge. While recommended items with regular prices only lead to a conversion rate of 0.45%, the rate for discounted items was at least 18 times higher with a value of 8.12%. This clearly indicates that recommending discounted items can be a promising strategy in this domain.

2.3 Systematic Feature Analysis

The statistical analysis from the previous section showed that certain item features (popularity and discount status) can be indicators for the success of recommendations. In this section, we report the results of a more comprehensive and systematic analysis, which should help us understand what makes a recommendation successful. To assess the importance of different potentially relevant factors, we first used the available data to frame a classification problem. Then, we applied different feature weighting methods to numerically estimate the importance of the factors. Each line of the resulting classification dataset corresponds to an item recommendation and is labeled as being successful or not. We then engineered 95 different features, including simple ones – like the popularity of an item during the last n days – as well as more complex ones that combine item characteristics with contextual or user-specific aspects[4]. An example for a more complex feature is the ratio of clicks by a user on items that have the same brand as the recommended one during the last n sessions. Since the dataset is very imbalanced and there are comparably few successful recommendations, we applied random downsampling to obtain an equal number of samples for each class.

Table 2: Gain Ratio

Feature	Weight
Discount level	0.439
Current popularity (day)	0.371
Discount flag	0.325
Viewed before	0.286
Current popularity (week)	0.242
Distance to first view (in days)	0.232
Distance to last view (in days)	0.217
Distance to first view (in sessions)	0.214
Distance to last view (in sessions)	0.210
Current popularity (month)	0.201

Table 3: Chi Squared

Feature	Rel. weight
Current popularity (day)	1.000
Current popularity (week)	0.785
Current popularity (month)	0.563
Discount flag	0.556
Discount level	0.556
Distance to last view (in sessions)	0.443
Distance to last view (in days)	0.441
Views count	0.435
Distance to first view (in days)	0.428
Distance to first view (in sessions)	0.428

Table 2 and 3 show the 10 most relevant features using the *Gain Ratio* and the *Chi Squared* method, respectively. For this measurement we sampled 2.000 *occasional* users and their successful recommendations from our dataset, leading to about 3.800 positive samples. The results of both methods are comparable and confirm the observations from the previous section. The most important features to predict the success of a recommendation are the recent popularity of an item and the fact that an item is on sale. Furthermore, when the user has recently viewed the item (multiple times), the chances are good that a recommendation will be successful. This result is also in line with the observation that reminding users of known items through recommendation lists can be useful.

A correlation analysis of the relevant features over all positive and negative samples showed that while most features are not related, a measurable correlation between the discount level and the current item popularity exists (0.47 for popularity/day, 0.34 for popularity/week). This suggests that discounts in this domain can have a positive effect on sales and confirms that recommending on-sale items can be valuable.

[3]The normalized score was computed by dividing the number of events (clicks and purchases) of an item by the maximum number of events recorded for an item in the dataset.

[4]A detailed list of all features is can be found at http://ls13-www.cs.tu-dortmund.de/homepage/sac2017-cosr

3. CONSIDERING TRENDS AND DISCOUNTS WHEN RECOMMENDING

Previous work [8, 10] has shown that considering short-term interests and recently viewed items can help to increase the accuracy of the recommendation algorithms. In the following, we will now aim to assess the value of incorporating popularity and discount information into a recommender.

3.1 Experimental Setup

We apply the offline measurement protocol proposed in [8, 10] to numerically assess the prediction accuracy of different algorithms. The protocol is based on time-stamped user interaction logs which are organized in shopping sessions. For each user, we split the time ordered list of sessions into training and test set where the latter one contains 20% of the user's purchases. The goal then is to predict every item an online visitor will purchase in each session of the test set in which a purchase was made. Since the consideration of short-term shopping intents is crucial for the e-commerce domain, we always reveal the interactions of the last p sessions including the current one that precede the tested purchase as proposed in [10]. Finally, we apply user-wise five-fold repeated random subsampling to account for random effects.

The evaluation was conducted on the two datasets described above (*heavy* and *occasional*), which we sampled from the raw data. As accuracy measures we use the hit rate and the Mean Reciprocal Rank, each at list length ten, i.e., *HitRate@10* and *MRR@10*. The latter measure not only counts for how many of the purchases in the test sessions the algorithm made a correct guess, but also considers the position of the "hit" in the list. Since we hide and predict each purchase event in the test data individually, Precision is proportional to the hit rate (Recall) and not reported here.

3.2 Algorithms

Baseline Algorithms.

We use the following collaborative filtering (CF) recommendation algorithms as baselines to assess the value of incorporating additional features in the process:

- BPR: Bayesian Personalized Ranking [12] is a learning-to-rank method designed for one-class (implicit feedback) collaborative filtering problems that optimizes an AUC-like performance criterion. The method has been shown to outperform other techniques like Item-to-Item CF as well as different matrix factorization approaches for this and similar problem setups [7, 8].

- C-CoOcc: A baseline method that uses patterns of item co-occurrences in the users' recent navigation histories to implement a functionality of the form "Users who bought ..., also bought ..." (see [10]). The co-occurrence patterns are then applied to the user's recent interaction history to make recommendations.

- C-KNN. This method takes a user's most recent interaction history as an input and looks for past sessions in the training set related to the same items. The cosine similarity is used to determine the distance between such sessions and the recommendations are ranked in a weighted scoring process as described in [10].

In contrast to the BPR method, both C-CoOcc and C-KNN take the user's most recent shopping intent as a contextual factor into account.

Reranking Schemes.

We implemented several schemes that take the relevance-ranked list of a baseline algorithm as an input and apply different strategies to adapt the ranking based on additional information, e.g., about item properties or current discounts.

- RV: This simple method, also from [8], called *Recently Viewed* assumes that recent user interest in an item is a good indicator for an upcoming purchase. The method recommends items from the user's interaction history in reverse order.

- FM: *Feature Matching* was described as a successful strategy in [8] and moves those items up the list that have characteristics similar to those that the user has recently inspected, e.g., the same color or brand. The ranking is determined by the number of item characteristics that were also found in the user's current session. The ranking of the baseline strategy is kept for items that have the same number of matching features.

- DR: *Discount Reranking* ranks items higher that are currently on sale. In our data, a limited set of discrete discount levels is available for each purchase, and the method ranks items with the highest discounts first. Again, the order of the underlying baseline technique is kept for items with the same discount level.

- RPOP: The *Recent Popularity* method computes a normalized popularity value for each item during a recent period. The popularity is determined by counting the interactions with the items in the entire user community. Here, we will report the popularity of the day of the purchase, as this was the strongest success indicator according to our analysis above. When determining the popularity value, we only counted interactions that happened *before* the purchase to be predicted on the same day.

- Hybrids: Since the different features according to our analysis in the previous section are often not strongly related, we tested a number of different combinations of the reranking strategies. In particular, we combined the well-performing content-based FM ranking strategy with the methods proposed in this paper.

In the next section, we will particularly focus on the value of combining methods that have shown to work well in [8] (FM and RV) with techniques that additionally leverage information about discounts (DR) and trends in the recommendation process (RPOP).

3.3 Results

Table 4 and 5 show the results for the *heavy* and *occasional* datasets. In both cases, the parameter p of the protocol, corresponding to the number of revealed recent sessions, was set to 6. Different parameter values did not lead to significantly better results. The best values for each column are printed with a grey background. The overall best values are printed in bold face.

Table 4: *Hitrate@10* and *MRR@10* results for the Zalando *heavy* user subset.

Dataset	Zalando *heavy* users subset					
Baseline	C-KNN		C-CoOcc		BPR	
Metric@10	HR	MRR	HR	MRR	HR	MRR
WR(RPOP,DR,0.5)-FM	**0.382**	**0.220**	0.262	0.121	0.225	0.100
RPOP-FM	0.361	0.187	0.233	0.103	0.216	0.096
DR-FM	0.316	0.177	0.242	0.120	0.168	0.094
RV-FM	0.306	0.097	0.266	0.096	0.262	0.111
FM	0.281	0.093	0.145	0.052	0.119	0.046
No postprocessing	0.268	0.091	0.123	0.046	0.062	0.021

Table 5: *Hitrate@10* and *MRR@10* results for the Zalando *occasional* user subset.

Dataset	Zalando *occasional* users subset					
Baseline	C-KNN		C-CoOcc		BPR	
Metric@10	HR	MRR	HR	MRR	HR	MRR
WR(RPOP,DR,0.5)-FM	**0.376**	**0.213**	0.225	0.115	0.273	0.129
RPOP-FM	0.359	0.192	0.212	0.108	0.277	0.132
DR-FM	0.275	0.152	0.197	0.095	0.206	0.111
RV-FM	0.296	0.097	0.198	0.076	0.244	0.097
FM	0.257	0.091	0.129	0.049	0.192	0.076
No postprocessing	0.241	0.087	0.109	0.043	0.152	0.060

Baseline Results.

The last rows of the tables show the results when *no postprocessing* is applied to the ranked lists returned by the baselines BPR, C-CoOcc, and C-KNN. For both datasets, C-KNN substantially outperforms the non-contextualized BPR method and the more simple contextualization method C-CoOcc on both measures.[5] This clearly shows that considering individual items of interest in the current browsing session *in isolation* (as C-CoOcc does) is not sufficient. C-KNN considers all recent interactions of a session and is thereby more effective in finding additional relevant items from past sessions. On the *occasional* dataset, even the BPR method, which only relies on long-term user models and no context information, is better than the C-CoOcc method. This indicates that the C-CoOcc method could not find a sufficient number of frequent co-occurrence patterns in this smaller dataset, which comprises the same number of users but has fewer interactions per user.

Adding Feature Similarity and Interaction Recency.

Moving up from the bottom-most row of the tables, we see that focusing the recommendations on items that have similar features as those inspected in the current session – as done by the FM strategy – helps to improve the hit rates and the MRR in every case. Also, considering the recency of item views as a ranking criterion (RV-FM strategy) leads to another accuracy increase as was shown in [8]. Note that RV-FM specifies a cascading hybrid strategy where the items are first pre-filtered based on their features and then re-ranked based on the time-stamp of the last interaction with it, in case the user has interacted with the item before.

Including mostly recently viewed items (reminders) in recommendation lists can help to achieve high prediction accuracy as shown in [10]. The recommendations might however be too obvious and prevent the recommender from pointing users to other potentially relevant items unknown to them. In the following, we will therefore focus on the value of recommending discounted and trending items.

Adding Discounts and Recent Popularity Information.

The effects of putting more emphasis on items that are currently on sale are given in the row DR-FM in Table 4 and Table 5. Across all configurations, ranking the items according to their level of discount is measurably better than ranking only based on their feature similarity (FM strategy).

A similar increase in prediction accuracy can be obtained by recommending recently trending items. The row labeled with RPOP-FM shows the results obtained when pushing items up the list according to their popularity on the day of the session in which the purchase was made. We varied the popularity time span to consider, e.g., the trending items of the week or month. The results show that recommending items that are trending over a longer period is also helpful; the strongest effects can however be obtained by looking at the most recent trends.

The success of recommending currently trending items can, according to our correlation analysis, at least in part be explained by the fact these items might be discounted. Nevertheless, since other factors might also increase the popularity of the offered items, we tested the hybrid strategy WR(RPOP,DR,0.5)-FM which combines popularity and discount information in a weighted approach.[6] This novel approach, in combination with the strongest baseline C-KNN, led to the *best overall results* for both datasets. The difference between the best-performing method and any other method was statistically significant at $p < 0.01$ according to a Student's t-test with Bonferroni correction.

[5]The C-KNN method was in similar form applied to the music playlist continuation problem in [1] and exhibited competitive performance also in this domain.

[6]The value 0.5 indicates that both aspects received equal weights, which was the best configuration in our tests.

Implications.

The analyses in Section 2 indicate that recommendations that are adopted by consumers are often related to trending and on-sale items. The results presented in this section show that these insights can be successfully operationalized in recommendation algorithms, e.g., through post-processing. At least for the fashion domain analyzed in this paper this in particular suggests that consumers do not consider items that are labeled as being discounted as potentially biased advertisements. In contrast, pointing consumers to on-sale items through recommendation lists seems to be an effective strategy, which, to our knowledge, has not been investigated in the recommender systems research literature before. An additional analysis showed that recommending on-sale items is not only effective for typical "bargain hunters", who have a tendency to mostly purchase discounted items. In contrast, discount recommendations were effective also for users for whom discounts did not play a role in the past.[7]

At the same time, recommending trending items has also proven to be effective in our experiments. At least to some extent, items became trending when they were put on sale, but not all trending items were actually on sale. Given the data that is available to us, we could not identify with certainty what caused the short-term popularity effects. It could be the result of a marketing campaign, it could be due to the recent addition of an item to the shop, or some other external event. We tested if some of the trending items appeared unproportionally often in recommendation lists, causing a "rich-get-richer" effect, which was however not the case. Also, the trending items were usually not the first items consumers looked at in a session, which would have indicated a deep link to the shop item from a marketing campaign. Nonetheless, a general strategy for a recommender in practice could be to constantly monitor the current popularity of the items and include (some of the) short-term bestsellers into the recommendation lists more often. Adding "a good dose of popularity" was also previously mentioned in [5] as a successful strategy for the domain of movie recommendations.

4. CONCLUSIONS

With our work we aim to contribute to a better understanding of what makes recommendations successful in practice. In this paper we focused on e-commerce recommendations and analyzed the recommendation logs of a large fashion retailer. The results show that besides the recommendation of recently viewed items, recommending on-sale items and currently trending items can lead to higher adoption rates of the recommendations in this domain. Our future work includes the analysis to what extent our findings generalize to other product domains and recommendation scenarios.

Generally, our work also shows that automatically identifying the factors that contribute to the success of recommendations by modelling it as a classification problem can help to generate better recommendations for a given application domain. The most important features in the fashion domain according to our analysis were recent item popularity, discounts, recent item views, and content-wise similarity to previously inspected items. The conducted experiments in [8, 10] and this paper showed that considering these aspects helped to improve the accuracy of our domain-specific algorithms. Generally, we therefore argue that a systematic analysis of successful recommendations can be one possible way to approach the largely open problem of predicting the online success of a recommender from offline experiments, as discussed, e.g., in [4, 5].

[7]More sales of discounted items usually leads to higher revenue but not necessarily more profit. These aspects have to be balanced by the service provider.

5. REFERENCES

[1] G. Bonnin and D. Jannach. Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys*, 47(2):26:1–26:35, 2014.

[2] P. Cremonesi, F. Garzotto, and R. Turrin. Investigating the persuasion potential of recommender systems from a quality perspective: An empirical study. *ACM TIST*, 2(2):11, 2012.

[3] K. Diehl, E. van Herpen, and C. Lamberton. Organizing products with complements versus substitutes: Effects on store preferences as a function of effort and assortment perceptions. *Journal of Retailing*, 91(1):1–18, 2015.

[4] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, and A. Huber. Offline and online evaluation of news recommender systems at swissinfo.ch. In *RecSys '14*, pages 169–176, 2014.

[5] C. A. Gomez-Uribe and N. Hunt. The Netflix recommender system: Algorithms, business value, and innovation. *ACM TMIS*, 6(4):13:1–13:19, 2015.

[6] D. Jannach and K. Hegelich. A case study on the effectiveness of recommendations in the mobile internet. In *RecSys '09*, pages 205–208, 2009.

[7] D. Jannach, L. Lerche, F. Gedikli, and G. Bonnin. What recommenders recommend - an analysis of accuracy, popularity, and sales diversity effects. In *Proc. UMAP '13*, pages 25–37, 2013.

[8] D. Jannach, L. Lerche, and M. Jugovac. Adaptation and evaluation of recommendations for short-term shopping goals. In *RecSys '15*, pages 211–218, 2015.

[9] E. Kirshenbaum, G. Forman, and M. Dugan. A live comparison of methods for personalized article recommendation at Forbes.com. In *ECML/PKDD '12*, pages 51–66, 2012.

[10] L. Lerche, D. Jannach, and M. Ludewig. On the Value of Reminders within E-Commerce Recommendations. In *UMAP '16*, 2016.

[11] J. Liu, P. Dolan, and E. R. Pedersen. Personalized news recommendation based on click behavior. In *IUI '10*, pages 31–40, 2010.

[12] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *UAI '09*, pages 452–461, 2009.

[13] Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-more Filtering. In *RecSys '12*, pages 139–146, 2012.

Investigating Personalized Search in E-Commerce

Dietmar Jannach
TU Dortmund, Germany
dietmar.jannach@tu-dortmund.de

Malte Ludewig
TU Dortmund, Germany
malte.ludewig@tu-dortmund.de

Abstract

Personalized recommendations have become a common feature of many modern online services. In particular on e-commerce sites, one value of such recommendations is that they help consumers find items of interest in large product assortments more quickly. Many of today's sites take advantage of modern recommendation technologies to create personalized item suggestions for consumers navigating the site. However, limited research exists on the use of personalization and recommendation technology when consumers rely on the site's *catalog search functionality* to discover relevant items.

In this work we explore the value of *personalizing search results* on e-commerce sites using recommendation technology. We design and evaluate different personalization strategies using log data of an online retail site. Our results show that considering several item relevance signals within the recommendation process in parallel leads to the best ranking of the search results. Specifically, the factors taken into account include the users' general interests, their most recent browsing behavior, as well as the consideration of current sales trends.

Introduction

Recommender systems are without a doubt one of the most successful applications of artificial intelligence technology that made its way from academic research to wide-spread industrial use. Automated recommendations are today a pervasive part of our online user experience and employed to recommend, for example, things to buy on e-commerce sites, friends to connect with on social networking sites, or content to consume on media streaming sites.

Recommendations can in general serve a variety of different purposes as discussed recently by (Jannach and Adomavicius 2016). In particular on e-commerce sites like Amazon.com, one key utility of recommendations is to help consumers find items of interest within large product catalogs more quickly, for instance, when the system displays items that are similar to the one the consumer is currently inspecting. Recommendations of this latter type – showing similar items – support different possible reasons why a consumer visits the site, namely "knowledge-building", "directed buying" and "search and deliberation", see (Moe 2003). While recommenders can also support other intents like "hedonic browsing" and item discovery, the work in this paper focuses on the goal-oriented reasons of a customer visiting a site.

Automated recommendations that are made in the context of a currently inspected item are however only one possible means to help consumers build up knowledge or understand the space of options. The typical functionalities implemented by most e-commerce sites include predefined, hierarchical catalog navigation structures and, as the focus of this work, a catalog search functionality.

Typical search engines on e-commerce sites allow users to change the order in which the search results are presented with pre-defined sort criteria. Common sort orders include "by sales rank", "by average customer rating", "by price", and often "by relevance", where the relevance function in this case is typically not further explained and might be a mix of relevance for the consumer and profit for the site.

Most of today's online shops do not offer an option for users to sort the results according to their past personal preferences or shopping behavior. It is however intuitive to assume that taking the consumer's past behavior into account when ranking the results can be useful, e.g., by ranking items up in the list that correspond to the consumer's typical price preferences, by listing products of the consumer's preferred brands first, or by promoting consumables that the user has already purchased in the past.

In this work, we aim to explore and quantify the potential value of ranking the search results in a personalized way. The underlying and obvious assumption is that the search functionality is more helpful for users when the more relevant items appear higher up in the result list. Technically, we will explore the use of different common recommendation techniques as well as additional problem-specific heuristics to personalize the search results for the individual consumer. To evaluate the effectiveness of the proposed techniques, we will use an offline experimental design using log data of a German online retailer of products for babies and small children and compare the results with existing approaches from the general field of web search personalization and recommendation.

The paper is organized as follows. Next, we describe our research methodology and the dataset used for the evaluation. Then, we will summarize the tested algorithmic approaches and report the results of the empirical evaluation. The paper ends with a discussion of related works.

Research Methodology

General Setup

The overall problem setting of e-commerce search personalization is in various ways similar to the problem of web search personalization. The typical inputs to a search personalization algorithm are a search string, a collection of documents, information about the past search behavior of individual users, and possibly additional context information. The computational task of algorithms is then to filter and rank the documents in a personalized way. Finally, measures like precision or recall can be used to compare algorithms.

Our research setup is slightly different in two ways. First, we consider item filtering to be a black box, i.e., we make no assumption about how the search strings are matched with the objects. The algorithms that we investigate in this work can therefore be applied in combination with any existing search component of an e-commerce site. Second, the main input provided to the algorithms for personalization is the recorded navigation behavior of users – data that is usually available in practical settings. The task of the algorithms is then to find the best possible ranking of a given set of objects and personalization therefore only affects the ranking of the items but not their selection.

Data and Ground Truth

The dataset that we will use in our research was provided to us by a major German online retailer of goods for babies and small children. The data comprises an anonymized subset of the navigation logs of the shop's web server collected over the period of one month in early 2016.[1]

Dataset Features The time-stamped log entries of the dataset are either marked as *page requests* or as *add-to-cart* events, which we use to determine purchase events. Each entry has an accompanying URL, which gives us more details about the requested page or the item that was added to the cart. By analyzing the URLs, we can further classify the page requests into the following relevant main categories:

- item view events,
- category browsing events,
- search events (including a search string),
- shopping basket checkout events.

Each log entry furthermore has a customer ID assigned, which is approximately derived from a tracking pixel. In addition, for each item we know some basic data like the category it belongs to, the brand, and the item's textual description.

Determining the Ground Truth In order to evaluate to what extent an algorithm is successful in generating a good ranking of the search results, we need a "ground truth" (gold standard) that defines whether or not an item is relevant for a search query or not. In our case, we consider an item as relevant and the search as successful whenever the consumer actually purchased an item that was returned by the search *in the same session*. Since the results returned by the site's engine are not available in the log data, we apply the following heuristic approach to re-construct a ground truth dataset from the log data.

1. For each search action in the logs, we repeated the search on the online shop through an automated agent.
2. The agent collected all results returned by the shop, using all available search orderings that were provided by the shop (e.g., by popularity, recency etc.).
3. We then inspected all navigation actions of a user after the search action in the same session[2]. Whenever a user actually *purchased* an item that was part of the search results in this session, we consider the search to be successful.

A successful search in this interpretation does not necessarily require that a user purchases an item *immediately* after inspecting the search results. If the user inspected various other items before buying one item in the same session, we still count the search as a (potential) success. Furthermore, we chose actual purchase actions as success indicators instead of item views, because the item view events can be biased by the ordering of the results, i.e., users typically click more often on the top-ranked items even though they are not necessarily the most relevant ones in the end. In that sense, our reconstruction heuristic is very conservative.

Evaluation Approach

Algorithm Task and Metrics At the end of the process of determining successful search actions, we can apply standard IR evaluation procedures. For each search query, we are given a set of items returned by the shop's search engine and the information which item of the set was actually purchased. Furthermore, we are given different alternative rankings (by popularity, by relevance etc.) provided by the shop and can apply the common measures hit rate (recall@n) and the MRR to determine the capability of an algorithm to rank the single relevant item at the top places of the result list.

The task of the recommendation approaches that we investigate in our research is to generate an alternative ranking of the items that were retrieved by the site's search tool. The problem of the generation of a candidate set is therefore not in the scope of our work. As a result, the algorithms that we analyze in this paper can be used independently of the internal mechanisms that are used by the site for retrieving relevant products for a given search query.

Data Sampling and Measurement Procedure As usual in e-commerce environments, we observe many users who have visited the shop only once and never made any purchase. To be able to apply personalization strategies, we created a subset of 5,000 active users of the shop for which we have at least 100 log actions and who have bought at least five items during the data collection period. Table 1 shows the characteristics of the resulting dataset.

We apply an evaluation protocol for session-based log data similar to (Jannach, Lerche, and Jugovac 2015). We first

[1] The data was sampled in a way that no conclusions about the visitors or the business numbers can be drawn.

[2] We split the log actions into sessions using a 30-minute period of inactivity as an indicator that a new session started.

Table 1: Characteristics of the resulting dataset

Number of users	5,000
Number of items	23,052
Number of purchases	42,905
Number of item view events	419,945
Number of successful searches	3,300
Avg. nb. of sessions per user	13.7
Avg. nb. of views per session	6.9
Avg. nb. of purchases per session	0.6
Avg. nb. of successful searches per user	0.66

split the time-ordered log data of each user into two parts. The most recent $20\,\%$ of the sessions containing successful searches form the test set and the other sessions are used as the training set. In the test phase, we therefore only consider users for which we have determined successful search sessions in the previous step. We iterate over all these successful search sessions in the test data, compute the personalized result rankings, and apply the above mentioned IR measures, which are at the end averaged across all tested users. To avoid random effects, we apply a five-fold user-wise cross-validation procedure. Note that using a sliding-window protocol over the time axis was not meaningful in our situation as we only have log data for one month.

Empirical Results

Compared Algorithms

The online shop from which we obtained the data provides four ways of ranking the results: by sales numbers, by some unknown relevance ranking, by price, and by the average consumer rating. The set of alternative ranking strategies used in our experiments is shown in Table 2.

We evaluate existing techniques from web search personalization, collaborative filtering, context-aware personalization strategies, as well as a heuristic that considers recent sales trends. For all algorithms we tuned their parameters in a manual process to optimize the hit rate for the complete dataset.

We have furthermore tested a variety of hybrid approaches to investigate their effectiveness and to illustrate the relative importance of the different aspects. We include the results of a limited selection (including the best performing one) of the various experiments in Table 3.

Evaluation Results

Table 3 shows the detailed results of our analysis using a dataset consisting of 5,000 users who frequently visited the site during the data collection period. We report the results for various configurations regarding the minimum number of search results. The ranking of the algorithms at the end was however very consistent across all configurations.

Configurations Since the effectiveness of different algorithms might depend on the number of existing search results for a query, we report the results for different configurations. The first row of Table 3 shows the threshold regarding the minimum number of results to be returned by the shop's search engine. The lowest threshold we consider is five, which means we do not consider search tasks in our evaluation that led to less than five results, because re-ranking these results which are all displayed on one single page might not add much value for the user. Considering higher thresholds can be informative in particular in the context of the reminding strategy, which might only be able to re-rank a smaller part of the search results as only a limited amount of past interactions with the items in the result set by the individual customer might exist.

The hit rates and MRR values obviously become smaller when the search results comprise more objects. The absolute values of the hit rates are generally comparably high, e.g., at around 0.4 for the most simple baseline method, because in many cases not too many results are returned and even a random recommender would place a number of relevant objects into the top 10 lists.

Performance of the Shop's Methods The best-performing method among the ranking strategies that are currently available on the site is the ranking according to the sales numbers, i.e., the *Bestseller* strategy. This strategy is however outperformed by any of the other algorithms tested in this work as shown in Table 3. We omit the detailed results for the other current ordering strategies that are implemented on the site.

Performance of Search Personalization Methods The *PClick* method, which is based on analyzing past successful searches, only works slightly better than the *Bestseller* strategy. The comparably weak performance of this strategy is most probably caused by the limited dataset size. Even though each of the top 25 search terms in our dataset was used more than 1,000 times by site visitors during the data collection period, there is a huge long tail of rarely used search terms and only 1% of the users entered one specific query repeatedly. Considering this fact, the PClick method often cannot make any valuable prediction and then defaults to the Bestseller strategy. We however believe that with a larger dataset, the performance of PClick would improve. While it might not be better than the "winning" strategies, it might represent a valuable component in a hybrid approach.

The content-based method (CB), which combines short-term and long-term models, works relatively well and outperforms, for example, the *C-KNN* method, whose recommendations are based on the user's short-term behavior. It is, however, not as accurate as the more elaborate content-agnostic learning-to-rank method BPR, which focuses on the user's long-term preference model. The somewhat limited performance of the CB method can in parts again be attributed to the comparably short data collection period and the fact that the long-term history is limited to one month. However, since user interests in the domain of baby goods might naturally change quite fast over time, this effect might be limited and focusing on more recent interactions might even be useful. Another aspect that limits the potential of the CB method are the often very short product descriptions. We have tested Doc2Vec as an alternative representation (Le and Mikolov 2014), which however led to even worse results.

Performance of Collaborative Filtering and Session-based Approaches The *C-KNN* method, which works very well in other recommendation scenarios, does not work as well as expected in the search personalization scenario. One reason could be that there are sessions which included a successful search but no other user actions. This happens when users arrived at the site, immediately searched for a specific item (e.g., a consumable) and directly proceeded with the purchase. In such situations, the available information might be too limited to find neighboring sessions.

The modern *BPR* method and the very simple, session-based *Feature Matching* (FM) technique lead to very similar results and *BPR* leads to slightly better MRR values, possibly due to the coarse-grained re-ranking strategy of the FM method. In sum, this observation corroborates existing insights, e.g., from (Matthijs and Radlinski 2011), that both long-term models (*BPR*) and short-term interests (*FM*) should be considered in search personalization.

Performance of Recommending Reminders and Trending Items Reminding users of items that they have recently inspected is the most successful individual strategy in our experiments. This indicates that users often browse items, which they later on – e.g., in the next session – retrieve again through the search functionality before making a purchase. Reminding users of known products in recommendation lists might not necessarily be the most valuable business strategy as reminders do not help consumers discover new areas of the item catalog. Nonetheless, reminders in search results help users locate their items of interest fast (e.g., consumables), which contributes to the usability of the site. In practical settings, one might consider to use hybrids

Table 2: Summary of Compared Result-Ranking Algorithms

Web-Search Personalization Techniques	
PClick	The rationale of this method is that users often search for the same things multiple times (Dou, Song, and Wen 2007). The method ranks those items up in the list that the user has clicked on in previous search sessions with the same or a similar search term.
Content-Based (CB)	An approach based on (Matthijs and Radlinski 2011) which relies on TF-IDF representations of the textual descriptions of the shop items. We compute weighted combinations of the mean TF-IDF vectors of the long-term and short-term models of each user and rank the search results based on the cosine similarity of the item descriptions and the user model.
Collaborative Filtering and Session-Based Recommendation	
BPR	A modern learning-to-rank collaborative filtering method designed for implicit feedback (Rendle et al. 2009), which we used to learn long-term interest models. We manually fine-tuned the parameters ending up with 150 features and 150 training iterations using an optimized learning rate and regularization factors.
C-KNN	This contextualized k-nearest-neighbor method recommends items from past shopping sessions that are similar to the most recent sessions of a user. Cosine similarity is used as a distance measure for binary item vectors (see (Lerche, Jannach, and Ludewig 2016)). The short-term profiling approach proved effective for e-commerce and also music recommendation in the past (Hariri, Mobasher, and Burke 2015).We systematically optimized the parameters. Using a neighborhood size of 10 and considering a user's last two sessions led to the best results.
Feature-Matching (FM)	This method proved effective for session-based recommendation in (Jannach, Lerche, and Jugovac 2015). It compares features (category, subcategory, and brand) of the items to rank with those of items the user inspected in the last session. Items that are a better content-wise match for the current session are ranked up.
Personalized Reminders and Global Trends	
Most-Recent Reminder (MR)	Reminding users of items that were of recent interest to them can be helpful (Lerche, Jannach, and Ludewig 2016), in particular when consumables are sold on the online shop. The MR strategy ranks items up (in reverse chronological order) that the user has interacted with in recent sessions. Considering the last 6 sessions for reminding proved to lead to the best results.
Trending-N	This strategy considers the last n days before the examined session and computes a normalized popularity score for each item to be ranked. Recently trending items are moved up in the search results according to this score. The best results were achieved when considering popularity trends of the last two weeks (Trending-14).
Hybrid Approaches	
HR1(Trending-14, FM)	A basic cascading hybrid which first ranks the items based on their recent popularity and then applies the feature matching method described above. The advantage of the method is that it also works in user cold-start situations and does not require computationally expensive long-term models.
HR2(Trending-14, FM, BPR)	A weighted hybrid which in addition considers the long-term user model using the BPR method. A grid search procedure was applied to determine optimal weights w.r.t. the hit rate. The final weights were 0.6 for Trending-14 and 0.2 for FM and BPR.
HR3(Trending-14, FM, BPR, MR)	An extension of HR2 which also includes reminders through the MR strategy. The parameters were again optimized through a grid search. The final values were 0.45 for MR, 0.4 for Trending-14, and only 0.075 for BPR and FM, which emphasizes the importance of reminders and popularity aspects.

Table 3: *Hit Rate@10* and *MRR@10* results for the dataset of 5,000 frequent users. The best values are printed in bold. Differences between the best performing method and the second best which are statistically significant according to the Wilcoxon signed-rank test ($\alpha = 0.05$) are marked with a star.

Min. nb. of result items	5		10		20		50	
Metric@10	HR	MRR	HR	MRR	HR	MRR	HR	MRR
HR3(Trending-14, FM, BPR, MR)	**0.685***	**0.394**	**0.675***	**0.382***	**0.636***	**0.363***	**0.584***	**0.324**
MR	0.671	0.389	0.652	0.371	0.619	0.352	0.572	**0.324**
HR2(Trending-14, FM, BPR)	0.634	0.315	0.624	0.304	0.565	0.282	0.511	0.241
HR1(Trending-14, FM)	0.626	0.306	0.604	0.289	0.564	0.267	0.504	0.235
BPR	0.605	0.297	0.579	0.284	0.535	0.262	0.477	0.226
FM	0.603	0.284	0.580	0.265	0.536	0.242	0.475	0.209
CB	0.560	0.273	0.535	0.258	0.487	0.238	0.416	0.198
C-KNN	0.555	0.268	0.524	0.246	0.480	0.227	0.403	0.185
Trending-14	0.537	0.229	0.506	0.208	0.455	0.185	0.375	0.148
PClick	0.476	0.194	0.438	0.172	0.385	0.150	0.321	0.127
Shop baseline (Bestseller)	0.467	0.191	0.433	0.168	0.380	0.147	0.314	0.125

which recommend both already known as well as new items.

Recommending items that have been popular within the last 14 days in an unpersonalized way proved to be a significantly better strategy than recommending the bestsellers of the entire data collection period ($\alpha = 0.05$)[3]. We found this strong difference somewhat surprising, given the short data collection period and this indicates that considering not only seasonable trends but also short-term spikes in sales (e.g., due to recent discounts) can be valuable for consumers.

Performance of Hybrids Combining a variety of different signals (long-term models, short-term models, reminders, and recent trends) in the weighted approach HR3 led to the overall best results and leaving out any of these components led to performance losses. In all except two cases the hybrid approach leads to statistically significantly better results according to the Wilcoxon signed-rank test as shown in detail in Table 3. *The main practical implication of our research therefore is that a multitude of signals should be considered in the search personalization process.*

The comparison of the strategies HR1 and HR2 shows again that including long-term preferences of the individual user *and* the user community is useful. Finally, even the simplest hybrid method HR1, which is computationally cheap and which works for cold-start users, was most of the time significantly better than all non-hybrid approaches (except for the reminders). This finding emphasizes the importance of considering both short-term trends as well as the visitors' individual and collective behavior for this problem setting.

Related Work

Our work draws on various insights from existing research in search personalization and recommender systems. While search result personalization for e-commerce sites, to our knowledge, has not been investigated in this form in the literature, the problem of web search personalization has been widely explored in the past. Typical strategies of such approaches include (a) the integration of personalization features in the ranking algorithm itself (Haveliwala, Kamvar, and Jeh 2003), (b) personalized query expansion (Chirita, Firan, and Nejdl 2007; Zhou, Lawless, and Wade 2012a), and (c) re-ranking in a post-processing step (Dou, Song, and Wen 2007). The algorithms evaluated in our work fall into this last category and we adopted a corresponding evaluation scheme used, e.g., in (Matthijs and Radlinski 2011).

Generally, the personalization and adaptation of search results is typically approached by deriving short- and long-term user profiles or by modeling the context of a search action from query, click-through, and other types of data (Bennett et al. 2015). Technically, this can be achieved with the help of weighted term vectors, ontologies, language models, and various machine learning techniques (Tan, Shen, and Zhai 2006; Sieg et al. 2007; Bennett, Svore, and Dumais 2010; Matthijs and Radlinski 2011).

In terms of additional data, existing research works investigated the following aspects: (a) the value of considering different contextual factors like the potentially underlying relationships between different search actions within a session or across multiple sessions; (Luxenburger, Elbassuoni, and Weikum 2008; Kotov et al. 2011), (b) the consideration of the user's current location (Bennett et al. 2011), and (c) the incorporation of signals from social media (Zhou, Lawless, and Wade 2012b; Carmel et al. 2009).

In addition, another family of approaches works by analyzing re-occurring search actions to adapt the ranking of the search results, e.g., (Dou, Song, and Wen 2007; Shokouhi et al. 2013).

In our experiments, we included two techniques from this field, namely a content-based approach that uses weighted term vectors as short-term and long-term models as well as one approach that learns from past successful searches. In theory, also other techniques can be applied, provided that the additionally required types of information like the user's social network activity are known.

In the field of recommender systems, session-based recommendation and the problem of balancing short-term and long-term interests was recently discussed, e.g., in (Jannach, Lerche, and Jugovac 2015). Similar to their work, we use BPR for learning a long-term model and different strate-

[3] We use the Wilcoxon signed-rank test ($\alpha = 0.05$) to assess the statistical significance of differences throughout the paper.

gies (C-KNN and FM) to incorporate short-term user interests. More elaborate techniques to capture the user's interests within a session, like proposed in (Hariri, Mobasher, and Burke 2012), could in principle also be applied to our problem in case sufficient information about the items is available. The value of including reminders within recommendations was discussed, e.g., in (Plate et al. 2006) and in (Lerche, Jannach, and Ludewig 2016). From the different strategies proposed in the latter work we included a simple yet effective method in our empirical investigations (MR).

Finally, there are different works that aim to extract useful insights from e-commerce search log data, e.g., (Duan et al. 2013) or (Liu et al. 2014), but their goal is often not centered around search personalization and focused, for example, on the unpersonalized diversification of the search results as done in (Yu et al. 2014). Personalized result rankings were discussed in the work of (Parikh and Sundaresan 2011), who however propose a manual approach, where users can interactively change the relevance of different sort criteria.

Conclusions

Our work showed that a variety of different factors should potentially be considered when personalizing search results in e-commerce. Since phenomena like short-term popularity trends and repeated item consumption are also common in other domains, including music or movies, we plan to investigate the use of hybrid approaches for search personalization in other fields as part of our future works.

References

Bennett, P. N.; Radlinski, F.; White, R. W.; and Yilmaz, E. 2011. Inferring and using location metadata to personalize web search. In *SIGIR '11*, 135–144.

Bennett, P. N.; Collins-Thompson, K.; Kelly, D.; White, R. W.; and Zhang, Y. 2015. Overview of the special issue on contextual search and recommendation. *ACM Trans. Inf. Syst.* 33(1):1e:1–1e:7.

Bennett, P. N.; Svore, K.; and Dumais, S. T. 2010. Classification-enhanced ranking. In *WWW '10*, 111–120.

Carmel, D.; Zwerdling, N.; Guy, I.; Ofek-Koifman, S.; Har'el, N.; Ronen, I.; Uziel, E.; Yogev, S.; and Chernov, S. 2009. Personalized social search based on the user's social network. In *CIKM '09*, 1227–1236.

Chirita, P. A.; Firan, C. S.; and Nejdl, W. 2007. Personalized query expansion for the web. In *SIGIR '07*, 7–14.

Dou, Z.; Song, R.; and Wen, J.-R. 2007. A large-scale evaluation and analysis of personalized search strategies. In *WWW '07*, 581–590.

Duan, H.; Zhai, C.; Cheng, J.; and Gattani, A. 2013. A probabilistic mixture model for mining and analyzing product search log. In *CIKM '13*, 2179–2188.

Hariri, N.; Mobasher, B.; and Burke, R. 2012. Context-Aware Music Recommendation Based on Latent Topic Sequential Patterns. In *RecSys '12*, 131–138.

Hariri, N.; Mobasher, B.; and Burke, R. 2015. Adapting to user preference changes in interactive recommendation. In *IJCAI '15*, 4268–4274.

Haveliwala, T.; Kamvar, S.; and Jeh, G. 2003. An Analytical Comparison of Approaches to Personalizing PageRank. Technical Report 2003-35, Stanford InfoLab.

Jannach, D., and Adomavicius, G. 2016. Recommendations with a purpose. In *RecSys '16*, 7–10.

Jannach, D.; Lerche, L.; and Jugovac, M. 2015. Adaptation and evaluation of recommendations for short-term shopping goals. In *RecSys '15*, 211–218.

Kotov, A.; Bennett, P. N.; White, R. W.; Dumais, S. T.; and Teevan, J. 2011. Modeling and analysis of cross-session search tasks. In *SIGIR '11*, 5–14.

Le, Q., and Mikolov, T. 2014. Distributed representations of sentences and documents. In *ICML '14*, 1188–1196.

Lerche, L.; Jannach, D.; and Ludewig, M. 2016. On the Value of Reminders within E-Commerce Recommendations. In *UMAP '16*, 27–25.

Liu, Z.; Singh, G.; Parikh, N.; and Sundaresan, N. 2014. A large scale query logs analysis for assessing personalization opportunities in e-commerce sites. In *WSCD Workshop at WSDM '14*.

Luxenburger, J.; Elbassuoni, S.; and Weikum, G. 2008. Task-aware search personalization. In *SIGIR '08*, 721–722.

Matthijs, N., and Radlinski, F. 2011. Personalizing web search using long term browsing history. In *WSDM '11*, 25–34.

Moe, W. W. 2003. Buying, searching, or browsing: Differentiating between online shoppers using in-store navigational clickstream. *J. of Cons. Psych.* 13(1):29 – 39.

Parikh, N., and Sundaresan, N. 2011. A user-tunable approach to marketplace search. In *WWW '11*, 245–248.

Plate, C.; Basselin, N.; Kröner, A.; Schneider, M.; Baldes, S.; Dimitrova, V.; and Jameson, A. 2006. Recomindation: New functions for augmented memories. In *AH '06*, 141–150.

Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI '09*, 452–461.

Shokouhi, M.; White, R. W.; Bennett, P.; and Radlinski, F. 2013. Fighting search engine amnesia: Reranking repeated results. In *SIGIR '13*, 273–282.

Sieg, A.; Sieg, A.; Mobasher, B.; Mobasher, B.; Burke, R.; and Burke, R. 2007. Web search personalization with ontological user profiles. In *CIKM '07*, 525–534.

Tan, B.; Shen, X.; and Zhai, C. 2006. Mining long-term search history to improve search accuracy. In *KDD '06*, 718–723.

Yu, J.; Mohan, S.; Putthividhya, D. P.; and Wong, W.-K. 2014. Latent dirichlet allocation based diversified retrieval for e-commerce search. In *WSDM '14*, 463–472.

Zhou, D.; Lawless, S.; and Wade, V. 2012a. Improving search via personalized query expansion using social media. *Inf. Retr.* 15(3-4):218–242.

Zhou, D.; Lawless, S.; and Wade, V. 2012b. Web search personalization using social data. In *TPDL '12*, 298–310.

User Modeling and User-Adapted Interaction

Session-based Item Recommendation in E-Commerce

On Short-Term Intents, Reminders, Trends, and Discounts

Dietmar Jannach · Malte Ludewig · Lukas Lerche

August 2017

Abstract Many e-commerce sites present additional item recommendations to their visitors while they navigate the site, and ample evidence exists that such recommendations are valuable for both customers and providers. Academic research often focuses on the capability of recommender systems to help users discover items they presumably do not know yet and which match their long-term preference profiles. In reality, however, recommendations can be helpful for customers also for other reasons, for example, when they remind them of items they were recently interested in or when they point site visitors to items that are currently discounted.

In this work, we first adopt a systematic statistical approach to analyze what makes recommendations effective in practice and then propose ways of operationalizing these insights into novel recommendation algorithms. Our data analysis is based on log data of a large e-commerce site. It shows that various factors should be considered in parallel when selecting items for recommendation, including their match with the customer's shopping interests in the previous sessions, the general popularity of the items in the last few days, as well as information about discounts. Based on these analyses we propose a novel algorithm that combines a neighborhood-based scheme with a deep neural network to predict the relevance of items for a given shopping session.[1]

Keywords Recommender Systems · E-commerce

1 Introduction

Automated and in many cases personalized recommendations of the type "You may also be interested in ..." are a common feature of modern e-commerce sites. Such recommendations can serve different purposes and create additional value for both customers and providers, e.g., by helping customers discover additional items of interest or by helping providers promote certain areas of their item spectrum.

The academic literature in the field mainly focuses on the recommendation utility for customers or the capability of algorithms to identify items that an individual user is presumably not aware of (Jannach and Adomavicius, 2016). In research settings the

D. Jannach, M. Ludewig, L. Lerche
Department of Computer Science, TU Dortmund, Germany
E-mail: firstname.lastname@tu-dortmund.de

[1] Parts of this work are based on (Jannach et al, 2015a), Lerche et al (2016) and (Jannach and Ludewig, 2017a). This paper or a similar version is not currently under review by a journal or conference, nor will it be submitted to such within the next three months. This paper is void of plagiarism or self-plagiarism as defined in Section 1 of ACM's Policy and Procedures on Plagiarism.

recommendation problem is often abstracted to one of matrix completion. The main algorithmic task given this problem formulation is then to reliably estimate the relevance of the unseen items based on the user's long-term behavior and preferences, and to create a ranked list of objects to be presented to the user.

However, in e-commerce environments, and in particular on retail websites that offer a wide range of different products, this research approach has a number of limitations and does not fully reflect all of the challenges of the domain.

- First, customers often visit e-commerce sites with a very specific shopping intent in mind (Moe, 2003). Relying solely on long-term preference models can be insufficient and algorithms have to adapt their recommendations to these short-term goals to be effective (Shani et al, 2005; Tavakol and Brefeld, 2014; Jannach et al, 2015a).
- Second, when relying on the matrix-completion problem formulation, the focus is on predicting the relevance of items for which no preference signal is given and which are presumably unknown to the user. However, many real-world systems also recommend items that the user has already seen or even purchased before, e.g., with the goal to remind users of things they were recently interested in or to provide convenient navigation shortcuts (Schnabel et al, 2016).
- Third, in many e-commerce domains, the purchase decision of a customer can depend on the current price of an item and some items might only be attractive if they are on sale. The relationship between prices and market demand has been extensively analysed in the economics literature for decades, e.g., based on statistical demand estimation approaches like the broadly-used one by Berry et al (1995), but it is not fully clear yet how price reductions impact the behavior of users in the context of recommendations.
- Fourth, an assumption in current research is that the recommendation of very popular items is of comparably little value, because the recommended items might be too obvious or of little novelty. Reports on real-world systems however indicate that recommending a mix of popular and lesser known items can be a good strategy (Garcin et al, 2014; Gomez-Uribe and Hunt, 2015).

Overall, whether or not an item should be recommended may depend on a number of factors other than its estimated match with the user's long-term preferences. These additional factors are however underexplored in the recommender systems literature. In this work, our aim is to systematically investigate the role of some of these factors – in particular those discussed above – and to design novel approaches to incorporate these factors into future recommender systems.

The structure and the contributions of this paper are as follows:

(i) In Section 2, we report the results of a systematic analysis of a large log dataset provided to us by Zalando[2], a European retailer of fashion products. A particular feature of the dataset is that it contains information about which items were recommended to users and which of these recommendations were successful, i.e., led to a purchase afterwards. This information allows us to systematically determine the characteristics of successful recommendations from log data, which to our knowledge has not been done before in the e-commerce domain.

(ii) Our analysis reveals that a larger fraction of the successful recommendations are actually *reminders*, i.e., items that the users have seen or purchased before. In Section 3 we therefore further elaborate on this topic and also report the results of a field test in a different e-commerce domain which shows that reminding users of known items can have a positive impact on the business.

[2] `http://www.zalando.com`

(iii) In Section 4 we finally show how the insights from the previous analyses can be operationalized within new recommendation algorithms. Specifically, we propose a hybrid method that combines a session-based nearest-neighbor scheme with a deep neural network that considers additional factors like discounts or recent sales trends in the ranking process. An experimental evaluation indicates that the proposed methods are more effective than previous approaches in terms of selecting and recommending items that are actually bought by customers on the site.

Generally, the work presented in this paper combines and extends a series of previous investigations on session-based recommendations in e-commerce environments. In (Jannach et al, 2015a) we mainly examined the relative importance of considering long-term preference models and short-term intents; in (Lerche et al, 2016) we focused on the value of placing reminders in recommendation lists; in (Jannach and Ludewig, 2017a) we finally designed a first recommendation method that used purchase prediction variables which were derived from a systematic analysis of log data. In this paper, we put these individual pieces together to provide a comprehensive picture of different phenomena that can be observed in practice but have not been explored in the literature to a large extent. Furthermore, we propose a novel method to combine the prediction features, which is more effective than our previous one and which is based on a deep learning technique.

2 Analyzing the Characteristics of Successful Recommendations

This section summarizes the main insights that were obtained through the analysis of Zalando's log data. The general goal of our analyses was to better understand in which cases the recommendations that were displayed to the users were successful (i.e., led to a purchase at the end).

2.1 Dataset Characteristics

The dataset provided to us contains a subset of the log of user interactions on the e-commerce site Zalando during about one year.[3] Actions were recorded for about 3.5 million users, which were identified through unique IDs based on cookies. The different types of actions that were made available to us include views of item detail pages, purchase actions, as well as add-to-cart and add-to-wishlist events. The recorded actions relate to more than 400,000 different shop items. For each item, we know various basic characteristics like the (anonymized) brand, the color, the item's catalog categorization (i.e., if it belongs to shoes or shirts), and the price level compared to other items of the same category. Furthermore, the log contains information about to which extent the item was discounted at the time of the user action.

Zalando's online shop also features a recommendation component, and additional items of interest are displayed to visitors when they navigate the site, for example, when they inspect the details page of an item.[4] Our log dataset contains the list of the top three recommended items that were displayed to the visitors on such pages. Click events on these recommended items were recorded as well, i.e., we know when a user visited an item through a link from the recommendation list. In our subsequent analysis, we consider a recommendation successful when the user purchased an item later on that he or she clicked earlier on a recommendation list.

[3] The dataset provided to us was fully anonymized and artificially distorted so that no inference on true sales numbers on the site can be made.

[4] Details about the used recommendation algorithm on the website were not disclosed to us.

Table 1: Characteristics of the Zalando datasets

	Raw dataset	*Frequent* users	*Regular* users	*Occasional* users
Users	3.5M	3,000	3,000	3,000
Items	460k	188k	121k	87k
Views	200M	3.1M	906k	452k
Purchases	3.9M	106k	47k	9.5k
Sessions	27.5M	338k	89k	64k
Sessions per user	7.79	113.11	29.91	21.31
Views per session	7.28	9.44	10.41	7.10
Purchases per session	0.14	0.31	0.52	0.15

The dataset is generally very sparse. Many users have visited the shop only once and a large majority of the users never made a purchase. Since one of our goals is to predict purchases based both on long-term and short-term user models, we focused on users that have interacted with the website several times in different shopping sessions and also made a minimum number of purchases. Removing non-buying users, which are not in the focus of our analysis, reduces the total number of unique customers by almost 80%, i.e., only about 760,000 of 3,5 million users remain.

For the various analyses and experiments made in this paper, we created a number of different data subsamples. Specifically, we varied the lower thresholds regarding the user's past activities, with the goal to assess if significant differences can be observed across different user groups, e.g., when applying different recommendation strategies.

- Our first sample contains 3,000 *frequent* (heavy) shoppers. We defined those customers as heavy users that had purchased at least 20 items during the year when the data was collected and who have interacted with the site within at least 40 sessions. On average, the randomly selected users of this group visited the online shop about two times a week, leading to an average of 114 sessions.
- The second sample covers a random selection of 3,000 *regular* shoppers, who purchased at least 10 items within at least 20 sessions. For the average customer in this group about 30 sessions were recorded, i.e., these users visited the site about every other week.
- The third sample comprises 3,000 randomly selected *occasional* site visitors, who interacted with the site at least in 10 sessions. No constraint on the number of purchases per user was applied for this dataset.

To further validate our novel prediction method presented in Section 4, we created the following additional datasets.

- A sample of 3,000 *random* users for which we only required that they purchased at least one single item. No constraint on the number of shop visits was applied.
- Two larger samples of *regular* users as described above, which comprise 5,000 and 10,000 users, respectively.

Table 1 shows some key characteristics of the three main datasets that we use for our analyses in the paper. The characteristics of the three additional validation datasets are given in Table 17 in Appendix D. Also in the appendix (Figure 6), we provide a visualization of the distribution of the purchase frequencies in the raw dataset. What we can for example observe in the data is that the frequent users visit the site more often, i.e., there are many more sessions per user, but their sessions are shorter on average and they make fewer purchases per session.

Generally, while the number of sampled users per dataset is comparably low, the resulting datasets still have a substantial size. The frequent user dataset for example

contains more than 3 million recorded item view events for almost 200,000 different items. We therefore see the dataset sizes not as a limiting factor of our research.[5]

2.2 Factors Influencing the Success of Recommendations

Based on our discussion of typical limitations of current research in the introduction of our paper, we investigated the impact of the following factors on the success of recommendations through different analyses. Each of these factors corresponds to one of the identified research limitations.

1. The importance of considering the users' short-term intents.
2. The effectiveness of recommending already known items.
3. The role of discounts.
4. The effects of considering (short-term) popularity trends.

We used the dataset of frequent shop visitors as a basis for the subsequent analyses. As a success measure for the recommendations, we calculated click-through rates and conversion rates. In our scenario, a *click-through* event happens whenever a user clicks on any of the items in a recommendation list and the click-through rate is computed as the number of click-through events divided by the number of page (recommendation list) impressions. We define a *conversion* as the situation when a user clicked on an item within a recommendation list and then actually purchased this item in the current or subsequent session. We determine the conversion rate by dividing the number of conversions by the number of clicks on any recommended item. Overall, the click-through rate therefore can be seen as an indicator to which extent the recommendations can attract clicks, whereas the conversion rate gives an indication of the effectiveness of a recommendation algorithm in terms of selecting items that match the users' preferences.[6]

In our sample of frequent users, we observed that about every 100^{th} displayed recommendation list received a click, i.e., a click-through rate of 1%. In general, the absolute value of the click-through rate can depend on various factors that are not related to the recommendation quality, including the visual layout of the web pages and the positioning of the recommendation list. Overall, a click-through rate of 1% shows that the displayed recommendations are not a central element for users to navigate the site.

The second and probably more interesting question is how often an item was added to the shopping cart and purchased after it was selected in a recommendation list, i.e., the conversion rate. In the dataset of frequent shop visitors, users placed an item into the shopping cart in one of the next two sessions in about 14 % of the cases when they had clicked on it within a recommendation list. In about 7 % of the cases when an item was clicked on in the list, it was actually purchased later on. This indicates that the recommender implemented on the site was in many cases able to select items that were truly interesting for the user. Note that this does not mean that only 7 % of the recommended items were interesting for the users, as we only know for the actually purchased items with certainty that they were relevant. In fact, other items shown in the recommendations might have been relevant as well, but not purchased at the end, e.g., because they were alternatives to the finally purchased item.

[5] Using much larger samples makes some of the experiments reported in later sections computationally challenging as some of the algorithms require extensive hyperparameter tuning.

[6] The conversion rate cannot directly be interpreted as the absolute amount of *additional* sales that is generated by a recommender since we cannot know if an item would have been bought by a user even when it was not recommended. Previous studies however indicate that recommenders in general can turn more visitors into buyers and be effective in terms of generating additional sales (Jannach and Hegelich, 2009).

2.2.1 The Importance of Considering the Users' Short-Term Intents

To assess how focused website visitors are when they arrive on the site, we analyzed their browsing behavior in terms of the diversity of the inspected items. An analysis based on the sample of frequent users showed that visitors on average look at the details of 9 different items per session and these items belong, again on average, to 2.7 different categories in the catalog. Given that there are more than 330 categories, this is a strong indicator that focused navigation behavior with a specific intent is common. Therefore, recommending mostly items that match the current shopping intent seems promising.

To quantify the importance of considering short-term intents when recommending, we calculated the recommend-to-purchase conversion rates for different situations: (a) when the recommended item was similar to the currently inspected one in terms of a certain feature (e.g., had the same color) and (b) when this was not the case. The results of this analysis are shown in Table 2.

Table 2: Recommend-to-Purchase Conversion Rates for Similar-Item and Different-Item Recommendations using the Dataset of Frequent Users

	Conversion rate when item feature has ...		
Item feature	... a different value	... the same value	Difference
Brand	0.950 %	4.227 %	345 %
Price level	1.403 %	3.624 %	158 %
Category	1.207 %	2.844 %	135 %
Color	1.521 %	2.701 %	77 %

The first row of the table for example shows that once a visitor arrived at the site and inspected an item of a certain brand, the conversion rate was 345 % higher when an item of the same brand was recommended than when items of other brands were recommended.[7] Much higher conversion rates are also achieved when items from the same price segment are recommended (158 %) and when the item belongs to the same category (135 %). Similarly, a substantial improvement (77 %) is observed when the recommendations have the same color as the last inspected item, even though the color scheme used by the shop is quite fine-grained.

Overall, the analysis shows that website users often have a comparably clear shopping intent when they visit the site and recommendations are correspondingly more successful when they relate to items that have characteristics that are similar to those of recently inspected ones. Therefore, recommending items from the most recent categories of interest might be a more effective strategy than making out-of-category recommendations. If the assumption holds that items of the same category are typically substitutes (alternatives) and items of other categories are complements (e.g., accessories), our observations would corroborate the findings of Diehl et al (2015) who also found a substitute-based store organization to be advantageous over a complement-based one for the fashion domain.

2.2.2 The Effectiveness of Recommending Already Known Items

When examining all three-item recommendations in the log, we noticed that about 10 % of these recommendations were items that the current user has inspected before at least

[7] Since brand loyalty is common in the fashion domain, a strong effect was generally expected.

once. The recommender system implemented on the site, like on other e-commerce sites such as Amazon.com, does therefore not limit itself to the recommendations of items that are presumably new to the user as is usually done in academic research.

The fraction of such *reminders* depends on the design and inner workings of the recommendation algorithm, which are unknown to us. The interesting part however is that nearly half (44 %) of the *successful* recommendations that finally led to a purchase were not new to the visitors. Obviously, we cannot know if the customers bought items *because* they were presented as reminders in the recommendation list. However, we can at least observe that visitors use the recommendations quite often as navigation shortcuts to items that they finally purchase (Plate et al, 2006; Schnabel et al, 2016). Including reminders in recommendation lists therefore seems to be promising in this environment.

On the other hand, the fact that more than half of the successful recommendations were actually unknown to the users shows that the implemented system is effective in helping users discover new items as well. From a practical perspective, this means that the most promising strategy could be to create recommendation lists that contain a well-balanced mix of already known items and items that are new to the visitor.

Generally, the selection of items that assumedly match the short-term interests can be made independently of the selection of the reminder items. However, once an estimate of the current user's shopping intent has been made, it can also be a plausible strategy to focus on reminder items that are a good match for the given shopping intent.

2.2.3 The Role of Discounts

In many domains, the price of an item has a direct impact on demand levels and sales. We could therefore hypothesize that items in recommendation lists that are marked as being discounted are more attractive to users and lead to higher conversion rates. On the other hand, there could also be a negative effect caused by recommending items that are labeled as being discounted. This could happen when visitors perceive the recommendations rather as advertisements than as unbiased hints by a benevolent recommendation system.

To analyze the effect in our dataset, we separately calculated the recommendation-to-purchase conversion rates for recommendations that were labeled as being on sale and items that were sold at the regular price. The observed differences were huge. While the conversion rates for non-discounted items were at only 0.45 %, the rate for items on sale was about 18 times higher and at 8.12 %. This is a clear indicator that on the analyzed website the recommendation of discounted items led to a higher effectiveness of the recommendation system.

Note that the majority of sales transactions during the data collection period were *not* involving discounted items. Furthermore, the existing recommender on the site on average only included one single discounted item in the recommendation list. Recommendations of discounted items were therefore disproportionately often successful.

2.2.4 The Effects of Considering (Short-Term) Popularity Trends

Recommending popular items is generally a "safe" strategy, even though it might not always lead to the highest business value as shown, e.g., in (Jannach and Hegelich, 2009). We made different analyses to identify a possible relationship between the popularity of a recommended item and the chances that the recommendation is adopted by a user.

A first analysis showed that whenever the visitor clicked on one of the three recommendations, the chances that it was the most popular one among the three were at 43 %, i.e., much higher than the theoretical 33 % random chance. For this particular measurement, we determined an item's general popularity by counting all view and purchase

events related to the item in the entire log dataset. However, in the fashion domain, seasonal trends are common and considering the item popularity over the period of one year is probably too coarse-grained. We therefore made an additional measurement in which we considered the popularity of the items in the recent past. As an example, we looked at how popular each item on a recommendation list was on the day on which the recommendation was made.

The results showed that recommendations were particularly successful when the recommended items were popular on that day and therefore probably represent recently trending items. Using a normalized popularity score, the average daily popularity of all recommended items was at 0.024, whereas the average of those which were actually selected afterwards was at 0.088, i.e., three times higher. The normalized score was computed by dividing the number of events (clicks and purchases) of an item on a specific day by the maximum number of events recorded for an item in the dataset on the same day.[8]

2.3 A Systematic Feature Importance Analysis

After having analyzed different potential success factors individually in the previous section, our next goal was to understand the *relative importance* of the different factors that can make a recommendation successful. To that purpose we used the available data to frame a classification problem where the task is to predict whether or not a displayed recommendation will later on lead to a purchase or not. Based on such a model, different feature weighting methods can be applied to numerically estimate the importance of the factors.

Correspondingly, each entry in the constructed classification dataset corresponds to an item recommendation recorded in the log data, which is labeled as being successful or not. We engineered a set of 95 different features as predictor variables. We included both comparably simple ones like the popularity of the item during the last n days as well as more complex ones that combine item characteristics with context-specific or user-specific aspects. An example for a more complex feature is the ratio of clicks by a user on items that have the same brand as the recommended one during the last n sessions. The full list of features can be found in Appendix C.

For the measurement, we again looked at the 3,000 *frequent* visitors and their successful recommendations from our dataset, leading to about 8,500 positive samples. Since the dataset is very imbalanced and there are comparably few successful recommendations, we applied random downsampling to end up with an equal number of samples for each class. Table 3 shows the 10 most relevant features using the *Gain Ratio* and the *Chi Squared* methods, respectively.[9] We used two alternative and popular feature selection methods to reduce the risk that our analysis is biased by the specifics of one single method. When looking at the most important features as listed in the table, we can observe that both methods lead to the exact same set of the 10 most relevant features (out of over 90). The order of the features is sometimes slightly different, which is caused by the specific ways the methods work.[10]

The results of both methods are therefore comparable and confirm the observations from the previous section. Every single feature in the top-10 list is either related to the recent popularity of the items, to current discounts, or related to the fact that a user has

[8] The popularity measurement only considered view and purchases related to the items up to the time point of the recommendation on that day.

[9] See, e.g., (Manning et al, 2008) for details about these feature selection methods.

[10] The results of the analysis for the set of 3,000 *occasional* users are similar and are reported in Appendix B. The detailed results of the feature analysis are provided at `http://ls13-www.cs.tu-dortmund.de/homepage/journal-cosr-2017`.

Table 3: Results of the statistical feature weight analysis.

Gain Ratio analysis	
Feature	Weight
Viewed before?	0.319
Any discount granted?	0.274
Discount level	0.274
Distance to last view (in days)	0.251
Current popularity (day)	0.249
Distance to last view (in sessions)	0.232
Distance to first view (in days)	0.199
Distance to first view (in sessions)	0.194
Number of previous views	0.181
Current popularity (week)	0.138

Chi Squared analysis (normalized)	
Feature	Weight
Current popularity (day)	1.000
Distance to last view (in sessions)	0.624
Distance to last view (in days)	0.619
Current popularity (week)	0.610
Number of previous views	0.603
Distance to first view (in sessions)	0.598
Distance to first view (in days)	0.595
Viewed before?	0.590
Any discount granted?	0.569
Discount level	0.569

recently viewed a given item in the past.[11] A correlation analysis of the relevant features over all positive and negative samples showed that while most groups of different features are not related, a measurable correlation between the discount level and the recent item popularity exists (0.47 for popularity/day, 0.34 for popularity/week). This could mean either that some items become particularly popular once they are discounted, or that the shop often reduces the prices for more popular items because discounts for popular items might help to attract more visitors to the site.

Overall, the analysis gives us a number of indications which factors contribute to the later success of a recommendation. In the following sections we will investigate how we can operationalize these insights when designing new recommendation algorithms.

3 A Detailed Analysis of Using Reminders Within Recommendations

The feature analysis results in Table 3 showed that there are in fact several features in the top ten list that are related to *reminders*, i.e., to items that the user has recently inspected. In this section we provide a more detailed analysis on the topic of reminders before we investigate the other features in Section 4.

Generally, reminders within recommendation lists have the unique characteristic that they do not help users discover so far unseen items, and it is therefore not fully clear to what extent they truly create business value. Our analyses so far only show that clicks on already known items in the recommendations lists comparably often lead to purchases afterwards. To assess the business value of reminders, we have performed an A/B test on an e-commerce site for electronics and gadgets in which we compared the effectiveness of a simple reminding strategy with other recommendation techniques. We will discuss the details in Section 3.1. Then, we will outline different alternative and more elaborate algorithmic approaches to select already known items for inclusion in the recommendation lists. The results of different offline experiments using the Zalando dataset will be summarized in Section 3.2.

3.1 Effectiveness Analysis: Results of a Field Test

To be able to conduct field tests on a real e-commerce website we collaborated with China-Gadgets[12], a German online retailer who specializes in consumer electronics and

[11] The entry "Distance to first/last view" in the table refers to the time that has passed since the user has viewed a recommended item the first or last time before the purchase.

[12] `http://china-gadgets.de`

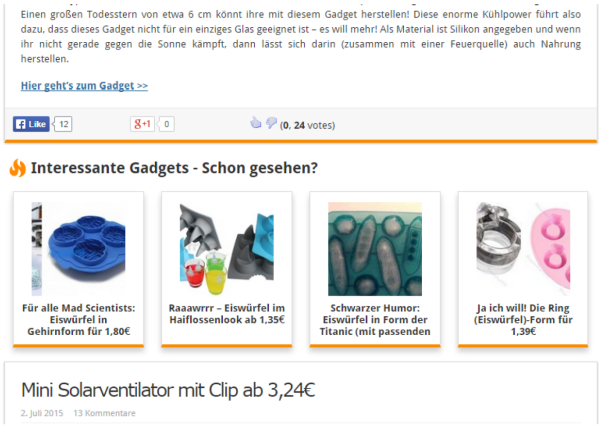


Fig. 1: Screenshot of four recommendations that were displayed below the most recent item of the China-Gadgets feed. In this example, the first item was a rubber ice tray to create ice "cubes" in the form of a Star Wars *Death Star* (not shown in the picture). The recommendations were created with the *Similarity* algorithm and are also rubber ice trays in the form of a brain, a shark fin, the Titanic, and a wedding band.

various types of gadgets. The front page of the China-Gadgets website is organized as a feed of product reviews, where the most recent item reviews are displayed at the top of the list. For each item, a details page exists which contains an in-depth review and a link to an external Chinese online shop.

3.1.1 Experiment Setup

For the purpose of the field study, we extended the existing website design and included recommendation lists at two places. First, we displayed four items below the most recent item, i.e., below the first entry of the feed, which can be seen in the screenshot in Figure 1. Second, we also showed recommendations on each item's detail page using the same layout and design as in Figure 1. Everything else on the website remained unchanged. We implemented the five strategies listed below in Table 5 to fill the recommendation lists.

Table 4: Dataset Characteristics

Characteristic	Value
Users	49k
Items	4k
Views	287k
Purchases	260k
Sessions	226k
Sessions per user	4.63
Views per session	1.94
Purchases per session	1.16
Popularity per item	134.96

During the experiment, which was conducted over the course of three months, the users of the website were randomly assigned to one of five conditions in an A/B test. Visitors were only once added to a group and the assignment was not changed when

Table 5: Brief descriptions of the algorithms used in the randomized field test.

Most Popular	A non-personalized baseline technique that recommends the most popular items of the categories to which the *reference item* belongs. The *reference item* is either the first item of the front page or the item of a detail page for which the recommendations are displayed.
Similarity	Another non-personalized method that recommends items that are "content-wise" similar to the reference item. The similarity between two items is determined by the angle between the TF-IDF-encoded representations of their plain-text item descriptions. We relied on the item descriptions because only a very limited set of item features was available to us.
Recently Viewed	A simple reminding strategy that displays the items that the current visitor has recently inspected in reverse chronological order.
Similarity Personalized	A personalized content-based method where the ranking of the items is determined by their distance to the current user's profile. The user profile is computed as the average TF-IDF vector of the items descriptions that the user has inspected in the past.
BPR	Bayesian Personalized Ranking (BPR) is a learning-to-rank collaborative filtering (CF) method for implicit feedback recommendation scenarios by Rendle et al (2009). To create a personalized item ranking for each user, a generic optimization criterion is optimized that is the maximum posterior estimator in a Bayesian analysis of the ranking task. Training of the model is done by utilizing a bootstrapped, stochastic gradient descent method. In the analysis of our previous work (Jannach et al, 2015a), BPR was the best-performing individual CF method. Contrary to the usual configuration of BPR, the algorithm is allowed to place items in the recommendation list that a user already knows. The BPR models were retrained every 30 minutes and the parameters of the method were optimized in an offline process beforehand.

a user repeatedly visited the site. All item view events and all clicks on links to the corresponding Chinese online shop were recorded.

To be able to measure the effects of personalization, we only considered users who visited the shop at least twice during the data collection period. The resulting dataset that we used in our analysis contained about 287,000 view events by about 49,000 users for over 4,100 products (see Table 4 for more details).

Since we also tested a popularity-based recommendation strategy, we analyzed the popularity distribution of the items (estimated by the number of view events). A distribution where a small set of popular items accumulates most of the view events would favor such a strategy. For the analysis, we grouped the items into 10 popularity levels with 400 items in each bin. Figure 2 shows the outcome of this grouping. The results reveal that the majority of the view events were indeed recorded for a comparably small set of items, and 10% of the items accounted for nearly 90% of all click events in the log data. Hence, there is a long tail of items that received very few clicks. The less popular half of all items, for example, were only involved in 1% of all interactions, with an average of 3.13 view events per item.

3.1.2 Observations

Looking at the general click-through rates across all conditions, about 2.6 % of the view events resulted from a click on an item in a recommendation list. The important business measure for China-Gadgets, however, is how often a visitor actually clicked on a link to the external Chinese retailer. In our analysis, we therefore consider a recommendation to be *successful* only when it was clicked by a user *and* when the user subsequently followed the link to the external website.

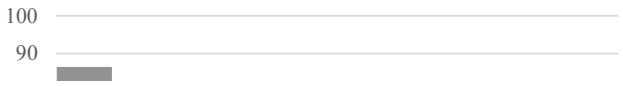


Fig. 2: Perc
in bins of 4

The res
to some su
when comp
cantly high
those that
popular ite
given that
comparably
strategy wa
be the resu
approach le

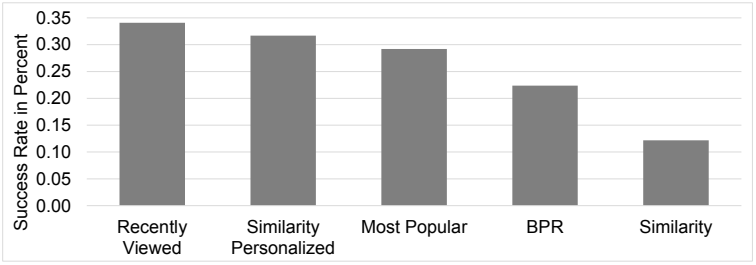


Fig. 3: Success rate of the different recommendation strategies in the China-Gadgets field experiment.

In contrast to typical recommender system evaluation setups, we configured all five algorithms in a way that they could also recommend items that the users already knew, i.e., for which we had observed item view events in the past. Across all configurations, we observed that 38 % of the *successful* recommendations were actually reminders, while only 20 % of *all* item suggestions were known to the users. In particular, the *Similarity*

[13] According to a four field Chi-squared test ($p < 0.05$) with Bonferroni correction.

Table 6: Percentage of reminders in the recommendations lists for all tested methods.

Method	Percentage of reminders
Recently Viewed	100.00 %
Similarity Personalized	82.56 %
BPR	24.95 %
Most Popular	3.33 %
Similarity	0.88 %

Personalized method tended to include a large number of known items in the recommendations (about 82 %) and these reminders are probably part of the success of the method in this experiment. Table 6 shows the percentage of reminders in the recommendations for all tested approaches.

Overall, our study suggests that including reminders in recommendation lists – besides serving as navigation shortcuts – can also be valuable from a business perspective. However, due to the site's structure, the results obtained in this field test have to be interpreted with care. One particularity of the China-Gadgets website, for example, is that new items are displayed in the form of a chronological news feed. For other online shops that follow a more traditional catalog and search-oriented website design, different effects may be observed.

3.2 Adaptive Reminders

The reminding method used in the field test proved to be quite effective despite its trivial nature. One can, however, imagine that simply recommending items in reverse chronological order might not be the best choice in all situations. Consider, for example, that a customer was searching for a pair of shoes and has – after inspecting a number of options – made a purchase in the last session. An intuitive approach when applying a reminding strategy would therefore be to *not* remind the user of shoes anymore in the next session. In the following sections, we will briefly summarize a number of such *adaptive* reminding strategies, which we originally introduced in (Lerche et al, 2016), and present results of offline experiments that were made using the Zalando *frequent* user dataset.

3.2.1 Proposed Strategies

In (Lerche et al, 2016), we proposed four general reminding strategies that have the goal to avoid too obvious recommendations. All strategies take a set of items as an input that the current user has recently interacted with, and re-rank and filter the items according to different goals. The strategies are briefly summarized in Table 7.[14]

Independent of the chosen reminding method, one possible additional strategy is to ignore an item when the user has recently made a purchase in the item's category. We call this the *Feature Filter (FF)* strategy because it filters out potential items to remind that have similar features as a recently purchased item. Technically, the *FF* strategy maintains a blacklist of categories for each user, and this blacklist is extended upon each purchase event. When the user, for example, has recently bought a pair of shoes, no reminders related to shoes will be displayed for some time. However, whenever a user continues to look for items that belong to a blacklisted category *after* the purchase, the category is removed from the blacklist again. Note that the *FF* strategy can be used in combination with any of the four re-ranking strategies proposed above.

[14] More information about the technical details of the algorithms are provided in (Lerche et al, 2016).

Table 7: Adaptive Reminding Strategies

Interaction Recency (IRec)	This method represents a generalized "Recently Viewed" strategy that considers also interactions other than item views (e.g., add-to-wishlist events). The items are ranked in reverse chronological order of the last interaction of the user with them.
Interaction Intensity (IInt)	In this method, the *amount* of recent interactions with a certain item is considered in addition to the last point in time of an interaction. More weight is given to items that the user has inspected more often.
Item Similarity (ISim)	The idea of this method is to remind the user of items that he or she inspected in a past session and which have similar features as the items that were inspected in the current session. If, for example, the user is inspecting shoes in the current session, the strategy ranks up other shoes that the user has inspected in the past. If the user has in the meantime looked up other types of products, e.g., scarves, those will be accordingly ranked lower, as they are from a different item category.
Session Similarity (SSim)	This method also considers the user's current session. In contrast to the *ISim* method, it does however not look for similar items, but for similar sessions. If a user in a past session was inspecting not only shoes but also matching belts, the *SSim* method will remind the user also of belts again, even though he or she has only started looking for shoes in the current session.

As a baseline to compare the reminding strategies with, we included a session-based recommendation method called *C-KNN* in our empirical evaluation. This nearest-neighbor algorithm compares the current session with all past sessions in the training data and then recommends items that appeared in past sessions that are most similar to the current one.

Technically, the algorithm works as follows. To speed up the similarity computations, we encode each session as a bit vector. Each vector element corresponds to an item and a "1" at a certain position means that an interaction with the item was recorded for the session. To quantify the similarity of the current session $\mathbf{c}$ with a historical session $\mathbf{h}$, we compute the cosine similarity $sim_{cos}(\mathbf{c}, \mathbf{h})$ between the vectors. Given the current session $\mathbf{c}$ we determine the k most similar sessions in the training data and denote this set as $mostSimilar(\mathbf{c})$. We then compute a ranking score $score(\mathbf{c}, i)$ for each item i given a current session $\mathbf{c}$ by summing up the similarity values of those sessions in $mostSimilar(\mathbf{c})$ that contain an interaction with i, i.e.,

$$score(\mathbf{c}, i) = \sum_{s \in \text{mostSimilar}(\mathbf{c})} sim_{cos}(\mathbf{c}, \mathbf{h}) \cdot \mathbf{1}_{\text{Interaction(s,i)}} \quad (1)$$

where $\mathbf{1}_{\text{Interaction(s,i)}}$ is an indicator function that returns true when item i appeared in session s. The recommendable items are finally ranked by the resulting scores in descending order.

The described method has proven to be very effective for session-based music recommendation in the past, see, e.g., (Hariri et al, 2012) or (Jannach et al, 2015b). According to additional experiments not reported here, *C-KNN* significantly outperforms more elaborate but session-agnostic methods like *BPR* for this problem setting.[15] The recommendations of the *C-KNN* method – in contrast to the other approaches tested in this setup – are not limited to already known items. This allows us to compare the effectiveness with non-reminding techniques on an absolute scale.

[15] We unfortunately were not yet able to benchmark *C-KNN* in a field study with our external partner.

3.2.2 Empirical Results

Using the generic evaluation protocol for session-based recommendations described in (Jannach et al, 2015a), we conducted a series of experiments on the Zalando and China-Gadgets datasets as well as on an additional dataset that was published in the context of the TMall recommendation competition[16].

According to our evaluation protocol, each recommendation algorithm is provided with a certain number of view events of the beginning of each session in which a purchase was made, and the goal is to predict the item that will be purchased. In addition to the views of the current session, the algorithm can be provided with information about the actions of the same user in the p preceding sessions.

We tested two different configurations of our protocol. The set of recommendable items consists of (a) either only those items that the user has interacted with in the p preceding session or (b) those that the user has interacted with in the p past sessions *plus* all item view events before the purchase in the current session. We call the latter configuration *reveal*, as the most recent views are revealed to the algorithm, and the other configuration *noreveal*. Since the algorithms are assumed to return ranked lists of items, we can use the standard information retrieval measures *hit rate* (i.e., recall given only one relevant item) and Mean Reciprocal Rank (MRR). Precision is proportional to recall in this setup.

Table 8 shows the main results of the empirical analysis for the Zalando frequent user dataset.[17] In this experiment, the possible items for recommendation (reminding) were taken from the last six user sessions ($p = 6$). In our previous work (Lerche et al, 2016), we experimented with different thresholds for the number of past user sessions from which reminders can be selected. Using too many previous sessions led to the inclusion of items that were already outdated. On the other hand, considering too few sessions makes the set of candidates that can be used as reminders too small. Overall, selecting $p = 6$ sessions led to the best results across the tested datasets including the Zalando dataset that is discussed here. In the *reveal* configuration in which the item view events of the current session were revealed to the algorithm, the absolute values, as expected, are generally higher. Users in almost all cases inspect an item before purchasing it. Purchases without item views in the same session only happen when the item was placed in the shopping basket already in a previous session.

Table 8: *Hit Rate@10* (HR) and *MRR@10* results for the Zalando frequent user subset. The best values for each configuration are highlighted in grey. The observations for the other datasets were similar, see also (Lerche et al, 2016).

Configuration	$p = 6$, reveal		$p = 6$, noreveal	
Metric@10	HR	MRR	HR	MRR
IRec	0.653	0.309	0.230	0.069
FF (*SSim*)	0.681	0.296	0.293	0.139
FF (*ISim*)	0.561	0.219	0.353	0.146
SSim ($k = 2$)	0.697	0.327	0.210	0.111
ISim ($k = 20$)	0.588	0.241	0.319	0.137
IInt	0.561	0.217	0.363	0.147
C-KNN	0.268	0.091	0.191	0.063

[16] The competition was organized in the context of IJCAI '15 and TMall is a leading Chinese online market place in the style of Amazon (`https://tianchi.aliyun.com/datalab/dataSet.htm?id=5`).

[17] For the sake of brevity we do not report the detailed results for the other datasets here. The additional results, which are in line with the observations for the Zalando dataset, can be found in (Lerche et al, 2016).

Overall, the results show that all reminding strategies are substantially and statistically significantly better in terms of the hit rate and the MRR than the best-performing baseline *C-KNN*:

- In the *reveal* condition, the "Recently Viewed" *IRec* configuration as expected is hard to beat as usually a view event is recorded some time before the purchase in the same session. Nonetheless, the results show that the *SSim* strategy, which looks for similar past sessions, can slightly outperform the *IRec* method.
- In the *noreveal* configuration, in contrast, the performance of the *IRec* and *SSim* strategies drops in comparison to the other techniques because in this condition they can only remind users of items that they have viewed in preceding sessions. The most effective strategy in this configuration is to consider how *often* an item was inspected by a user in the recent history (*IInt*). The feature filtering method *FF* also proved to be effective in this condition.

3.3 Discussion

The analyses in this section not only confirm the observation made earlier in (Jannach et al, 2015a) that including reminders in recommendations leads to higher accuracy values in offline experiments, but also that it can lead to increased business value in real-world applications. The analyses of the more elaborate strategies from the previous section furthermore indicate that better approaches than just recommending the most recently viewed items in reverse order exist.

How many items of a recommendation lists should be reminders and which strategy one should use to select the reminders largely depends on the specifics of the domain or application. In fact, our observations in Section 2 for the Zalando dataset showed that more than half of the successful recommendations were *not* reminders, so the problem exists to find a balance between the recommendation of already known items and novel item suggestions in practice.

4 Algorithmic Approaches to Improve Session-Based Recommendations

Having analyzed the different success factors for recommendations *ex post* in Section 2, our goal is now to operationalize these insights into new session-based recommendation algorithms that are better able to *predict* the next purchase than previous techniques.

4.1 General Approach – A Two-Phase Item Scoring Method

Our previous works in the field of session-based recommendations showed that considering both long-term preferences patterns and immediate short-term shopping intents can be key to high prediction accuracy in e-commerce scenarios (Jannach et al, 2015c). In the following, we will therefore adopt the same general approach and apply a two-phase item selection and ranking strategy:

- In the first phase, we use different baseline algorithms to compute an initial ranked set of recommendable items. The scoring is either based on long-term preferences only or based on a session-based scheme.

– In the second phase, we post-process the resulting sets and apply a number of re-ranking strategies, which consider the additional success factors for recommendations that were identified in Section 2, including discounts or trends in popularity.[18]

4.2 Baseline Scoring Techniques

We evaluated three baseline scoring techniques in our experiments.

– *BPR*, as it was the most effective context-agnostic baseline method according to the experiments in (Jannach et al, 2015c). Technically, the BPR model is learned on the long-term training data and the recommendations for a given session of the test dataset are only based on the ID of the current user without incorporating further contextual information.
– *C-KNN*, the session-based nearest-neighbor method mentioned above, because it led to the best accuracy values when additional factors like discounts were not considered.
– *C-CoOcc*, a session-based method that computes recommendations of the type "Customers who bought ... also bought ...". Technically, this method determines pairwise item co-occurrence patterns (association rules of size two) in the training data and then recommends those items that most frequently co-occur with the items in the current session. We include this method to assess to which extent our re-ranking methods lead to improvements when a simpler but commonly used baseline technique is employed.

4.3 Heuristic Re-Ranking Strategies

In (Jannach and Ludewig, 2017a) we proposed first approaches to incorporate additional relevance signals into the ranking process. The approaches were based on comparably simple heuristics for each of the potentially relevant success factors (short-term intents, reminders, trends, and discounts) identified earlier in this paper. A brief summary of these heuristic approaches is provided in Table 9.

Table 9: Heuristic Re-Ranking Strategies

Feature Matching (FM)	This strategy takes the user's short-term preferences into account by ranking those items up (given a baseline-ranked set of recommendations) that have common features with items that the user has inspected in the current session. Items with more matching features are ranked higher. This simple scheme proved to be effective according to the analyses in (Jannach et al, 2015a).
Interaction Recency (IRec)	This method, as described above, simply places those items in front of the list that the user has recently inspected. The items are sorted in reverse chronological order (most recent ones first). Items that were not viewed recently by the user are subsequently ranked according to the baseline score.

[18] In our empirical evaluation presented in Section 4.5 the baseline algorithms were used to create a set of 200 recommendations to be re-ranked, which on average led to the best results in terms of the hit rate.

Recent Popularity (RPOP-n)	This strategy considers recent sales trends in the ranking process. We compute a normalized popularity score for each recommendable item for the last n days. To compute the score we simply compute all interactions (views, purchases) of all users with each item in the last n days. For the last day, i.e., the one of the current session, we only count interactions up to the time point when the session started. In the following, we will report the results obtained for *RPOP-1*, i.e., we only consider the popularity of the current day, because this led to the best results according to our experiments.
Discount Promotion (DP)	This method ranks up items that are currently on sale. In our dataset, each purchase event can have one of four discrete *discount levels* assigned. Items with the highest (relative) discount are ranked up. Items with identical discount levels are ranked according to the baseline score.

Our systematic analysis in Section 2 has shown that most of the success factors are not correlated. We therefore implemented and evaluated a number of *weighted hybrid* strategies, which combine the outcomes of the four heuristic re-ranking strategies (*FM*, *IRec*, *RPOP-n*, *DP*). Through a series of experiments, as reported in (Jannach and Ludewig, 2017a), we determined the following mix of a cascading and weighted combination involving three heuristic strategies as the most effective one:

1. We apply the *FM* strategy to retain only items that share at least one feature with any of the items viewed in the most recent sessions.
2. We then calculate the *RPOP* and *DP* scores for these items.
3. The two scores are finally combined in a weighted approach and used for re-ranking.

We denote this combination as *WR(RPOP,DP,0.5)-FM*, where the numerical parameter indicates the relative importance of the two scores returned by *RPOP* and *DP*.

4.4 A Classification-based Approach Using Deep Neural Networks: DeepPredict

A main disadvantage of the presented heuristics-based approaches is that the re-ranking scheme is partially coarse-grained and that a lot of fine-tuning is required to find the right combination of heuristics and the corresponding weights when hybrids are used. We are therefore interested in developing a learning-based technique which is able to consider a variety of signals in parallel and which is able to determine optimal weights automatically.

In this section, we propose such a learning-based approach, which we call DeepPredict. The general approach is similar to the one applied in Section 2 and is based on using the available log data to frame a classification problem. In contrast to Section 2, however, the setup is slightly different. In Section 2, we were interested in determining the *general* importance of different features as predictors for the success of a given recommendation. The goal this time, however, is to predict which item will be purchased given the user's *specific and usually multiple* interactions in the current and previous sessions. Therefore, the problem encoding is different from the one used in Section 2 and is based on slightly different features as will be described below.

The outcomes of the classification task are numerical scores that express the probability that the user will click and purchase a recommended item. In contrast to traditional classification problems, we however do not use these scores to classify the recommendable items but use them to rank items that were identified as possible recommendations by a baseline algorithm, e.g., *C-KNN*.

4.4.1 Feature Engineering, Problem Encoding, and Sampling

Features. As a consequence of the slightly different problem setup, many of the features that we introduced in Section 2 cannot be used anymore. For example, we cannot any

Table 10: Features used in the learning-based approach

Feature	Description	Nb.	Type
Popularity	Normalized popularity of the item in the same day, week, or month	3	Float
Viewed before	True, if the item was viewed before by the user	1	Bool
Views count	Number of previous views of the item by the user	1	Integer
Distance to first view	Distance to the first item view by the user in days or sessions	2	Integer
Distance to last view	Distance to the last view of the item in days or sessions	2	Integer
Brand ratio	Fraction of items of the same brand in the last 1, 2, and 3 sessions	3	Float
Brand popularity	Overall popularity of the brand for the same day, week, or month	3	Float
Color ratio	Fraction of items with the same color in the last 1, 2, and 3 sessions	3	Float
Color popularity	Overall popularity of the color for the same day, week, or month	3	Float
Category ratio	Fraction of items of the same category in the last 1, 2, and 3 sessions	3	Float
Category popularity	Overall popularity of the category for the same day, week, or month	3	Float
Price level ratio	Fraction of items of the same price level in the last 1, 2, and 3 sessions	3	Float
Discount granted?	True, if the item is discounted	1	Bool
Discount level	Level of discount from 0 (no discount) to 3 (high discount)	1	Float

longer compare an item's color with the color of the *currently viewed item* as a binary feature. In fact, in the new problem situation we now have to consider a session-based context with *multiple items of different colors*. We therefore reduced the set of all features from Section 2 to 32 session-based features for the task of predicting the probability of whether or not a certain item will be purchased in the current session.

The set of features that were used in our experiments is shown in Table 10. A substantial number of these features is domain-specific, including for example those related to the color or the brand of an item, which are particularly important in the fashion domain. In general, however, the proposed classification-based approach can be applied for arbitrary domains and other domain-specific features can be included if needed.

Problem encoding and sampling. Differently from the encoding in Section 2, a positive training example is no longer based on the information if a certain item was shown in a recommendation list and then purchased. Instead, a positive training example is derived from the log data for each view event for which we later observed a purchase of the same item in the same or the next session. A view event that did not lead to a purchase is, in contrast, encoded as a negative example.

Since there are far more negative training examples than there are positive ones, for each positive example found in the user sessions we randomly sample exactly one negative example to end up with a balanced dataset.

4.4.2 Model Selection, Optimization, and Application

Model Optimization and Selection. After the feature engineering phase, a set of labeled training examples can be derived from the training part of the user log data.[19] Given

[19] In the experiments reported below we used the *frequent* user sample, as described in Section 2.1, as a basis for learning.

such a dataset, a variety of different supervised machine learning methods can in principle be applied. Using the RapidMiner software suite[20], we tested a number of different techniques, including classification using decision trees, random forests and logistic regression. To determine the best-performing algorithm and its parameters, we systematically searched for the best configuration by applying grid search in combination with ten-fold cross-validation, using again the functionality provided by the RapidMiner software. Furthermore, we applied a genetic feature selection strategy to find an optimal set of features for the given algorithm configurations.

At the end, the best model fit was achieved when testing an optimized artificial neural network with 2 hidden layers, each with 50 nodes, using the implementation of the H_2O open source artificial intelligence library[21], which is available as an add-on component for RapidMiner. We briefly explain the neural network in the next section. The feature selection process for the neural network based approach led to minor improvements when seven of the 32 features were excluded.[22]

A random forest classifier led to the second best accuracy scores (with 23 trees and a maximum depth of 49). For comparison purposes, we accordingly conducted experiments where we re-ranked the recommendations by the scores returned by this classifier. We call this alternative method RFPredict. In the case of random forests, feature selection did not lead to further improvements, which is probably caused by the inherent characteristic of the algorithm to implicitly include only important features in the trees, depending on the number of features and the configured depth of the trees.

Deep Neural Network Encoding The deep learning algorithm provided by H_2O is based on classic feedforward neural networks consisting of multiple layers of multiple nodes, each representing a human neuron as a weighted sum of inputs with a bias b as shown in Figure 4. All these nodes are layer-wise interconnected and the neurons' outputs are transformed with a nonlinear activation function f. In our network (see Figure 5), the

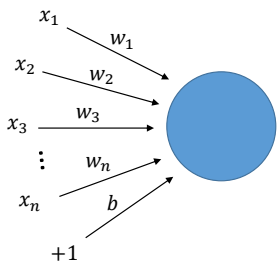


Fig. 4: Neurc

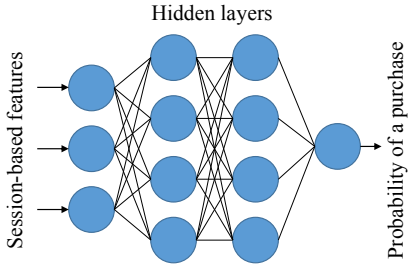


Fig. 5: Network

All weights and biases were optimized in a supervised learning process to minimize the difference between the predictions of the model and the real labels in our examples. The activation function that led to the best results in our model optimization process was the linear rectifier $f(x) = \max(0, x)$ combined with cross entropy

[20] `https://rapidminer.com`

[21] `https://h2o.ai`

[22] The features that were not considered by DeepPredict are listed in the appendix in Table 16. All non-considered features are related to the recent popularity of a certain brand or category.

($L(y,p) = -y\log(p) - (1-y)\log(1-p)$) as the loss function, which is commonly used for binary classification problems.[23] For the optimization of the network, the approach implemented in H_2O applies a parallelized stochastic gradient descent (SGD) method with back-propagation, see (Goodfellow et al, 2016; Candel et al, 2017). Furthermore, the SGD algorithm incorporates different optimizations like regularization, momentum, and an adaptive learning rate (ADADELTA) as discussed by Hinton et al (2012), Zeiler (2012), Sutskever et al (2013), and Wager et al (2013).

Model Application / Recommendation. As was done with the heuristic re-ranking approaches presented in Section 4.3, we applied the neural network-based method DeepPredict and the random forest-based method RFPredict in the form of a post-processing strategy in the recommendation phase.

1. Again, we use different baseline algorithms (e.g., *C-KNN*) to compute an initial ranked set of recommendable items.[24]
2. We consider each of these items as an unlabeled example and generate values for all features from Table 10 given the specific situation of the current session. Given an item to re-rank, we for example compute the fraction of items in the last two sessions that had the same color.
3. Finally, we apply our pre-trained model to all examples to predict if the corresponding item will be purchased in the current session by the user or not. The top items that were returned by the baseline algorithm are then (re-)ranked according to the probability score returned by the classifier.

4.5 Empirical Evaluation

We evaluated the different recommendation strategies on the three subsets of the Zalando dataset, i.e., the frequent, the regular, and the occasional user dataset, again using the evaluation protocol from (Jannach et al, 2015a), and hit rate and MRR as accuracy measures. The reported results are the average after applying a five-fold repeated-subsampling cross-validation scheme. As done in the previous experiments, we set the number of preceding user sessions to six ($p = 6$) and revealed all item views before the purchase in the current session.[25]

The obtained results for the datasets of frequent, regular, and occasional users are shown in Table 11, Table 12, and Table 13, respectively. The best accuracy values for each evaluated baseline are highlighted with a grey background and the overall best result per metric is emphasized in bold font.

4.5.1 Comparison of Baseline Strategies

The performance results of the individual baseline ranking schemes *BPR*, *C-KNN*, and *C-CoOcc* are shown in the first row of each table ("No post-processing")[26]. The *C-KNN* method, as already mentioned in Section 3.2.1, leads to the highest hit rate and MRR among all other baselines on all datasets.

[23] y is the correct label (1 or 0) and p the predicted probability.

[24] As in our previous experiments, we took the first 200 recommendations of the baseline algorithms as a basis for re-ranking as this cut-off value on average led to the best results in terms of the hit rate.

[25] Revealing fewer view items of the current session leads to accuracy values that are lower on an absolute scale; the ranking of the algorithms is however not changed.

[26] All reported differences are statistically significant according to a Student's t-test at $p < 0.01$ with Bonferroni correction.

Table 11: *Hit Rate@10* and *MRR@10* results for the Zalando *frequent* user subset.

Baseline	C-KNN		C-CoOcc		BPR	
Metric@10	HR	MRR	HR	MRR	HR	MRR
No post-processing	0.268	0.091	0.123	0.046	0.062	0.021
FM	0.281	0.093	0.145	0.052	0.119	0.046
IRec-FM	0.306	0.097	0.266	0.096	0.262	0.111
DR-FM	0.316	0.177	0.242	0.120	0.168	0.094
RPOP-FM	0.361	0.187	0.233	0.103	0.216	0.096
RFPredict	0.381	0.248	0.274	0.150	0.241	0.119
WR(RPOP,DR,0.5)-FM	0.382	0.220	0.262	0.121	0.225	0.100
DeepPredict	**0.405**	**0.284**	0.322	0.205	0.301	0.188

Table 12: *Hit Rate@10* and *MRR@10* results for the Zalando *regular* user subset.

Baseline	C-KNN		C-CoOcc		BPR	
Metric@10	HR	MRR	HR	MRR	HR	MRR
No post-processing	0.241	0.087	0.109	0.043	0.152	0.060
FM	0.257	0.091	0.129	0.049	0.197	0.077
IRec-FM	0.296	0.097	0.198	0.076	0.245	0.104
DR-FM	0.275	0.152	0.197	0.095	0.206	0.110
RPOP-FM	0.359	0.192	0.212	0.108	0.280	0.133
RFPredict	0.367	0.216	0.218	0.121	0.302	0.174
WR(RPOP,DR,0.5)-FM	0.373	0.213	0.225	0.115	0.316	0.155
DeepPredict	**0.389**	**0.253**	0.243	0.155	0.316	0.183

Table 13: *Hit Rate@10* and *MRR@10* results for the Zalando *occasional* user subset.

Baseline	C-KNN		C-CoOcc		BPR	
Metric@10	HR	MRR	HR	MRR	HR	MRR
No post-processing	0.405	0.197	0.134	0.053	0.142	0.055
FM	0.430	0.201	0.155	0.060	0.211	0.085
IRec-FM	0.501	0.217	0.235	0.090	0.296	0.125
DR-FM	0.472	0.226	0.209	0.094	0.261	0.137
RPOP-FM	0.501	0.241	0.240	0.118	0.321	0.149
RFPredict	0.500	0.295	0.264	0.159	0.356	0.206
WR(RPOP,DR,0.5)-FM	0.520	0.295	0.263	0.136	0.366	0.186
DeepPredict	**0.532**	**0.322**	0.285	0.177	0.356	0.212

On the dataset of frequent users, the context-aware but comparably simple *C-CoOcc* method significantly outperforms the context-agnostic *BPR* method, which only considers the user's long-term preferences. For the datasets consisting of regular and occasional users that in general have a shorter purchase history, in contrast, the *BPR* method works better than the *C-CoOcc* method.

In principle, one would expect *BPR* to work better for the frequent user dataset because more detailed profiles for each user are available. However, in the considered domain, some of the preference signals might be already outdated at the time of the recommendation. Since *BPR* has no built-in means to give less weight to older interactions, the learned model might focus too much on old interactions. For the dataset of regular or occasional users in contrast, the danger of overfitting to past interactions is less pronounced as the user profiles are less sparse in the first place. In such situations, *BPR*'s tendency to focus on popular items might in addition help to increase the obtained accuracy values, see also (Jannach et al, 2015c).

4.5.2 Considering Short-Term Intents, Reminders, Discounts, and Trends

In the next four rows of Table 11, Table 12, and Table 13, we show the effects of considering additional factors with the help of the heuristic re-ranking strategies presented in Section 4.3. The results show that each individual signal can measurably contribute to an accuracy increase.

(i) Applying Feature Matching (*FM*) as a post-processing strategy improves the accuracy results for every baseline and for all datasets. In the next set of measurements, we therefore always apply the *FM* method on top of the baseline ranking.
(ii) Adding reminders on top of the baselines and the *FM* strategy (*IRec-FM*) consistently leads to further increases in terms of the hit rate and the MRR. A similar effect was reported in (Jannach et al, 2015a) for different baselines. Nonetheless, while increases in accuracy can be achieved through reminders, focusing too much on reminders can be of limited business value, as such recommendations are not suited to point users to unknown but relevant items.
(iii) Next, we examine the effect of focusing the recommendations on items that are currently on sale (*DR-FM)*. The results show that this strategy proves effective and significantly more accurate when compared to the *FM* baselines across all the configurations. The *DR-FM* strategy is however not consistently better than the *IRec-FM* reminding strategy for all datasets and accuracy measures.
(iv) The *RPOP-FM* strategy, which focuses on recent trends, finally leads to the best accuracy results so far. It significantly outperforms all other methods that combine Feature Matching with one additional signal on the *regular users* dataset. It is also better for the *frequent users* dataset, except when the *C-CoOcc* strategy is used as a baseline. In this measurement, we report the results of the *RPOP-1* strategy, i.e., we only consider the popularity of the items on the day when the recommendations are made. Considering longer-term trends is also helpful, but the best results in terms of the hit rate and MRR were achieved when only the last day was considered.

4.5.3 Leveraging Multiple Signals in Parallel

We tested various weighted combinations of our heuristic re-ranking approaches. Our analyses in Section 2 showed limited correlations between most success indicators, and we in fact achieved the best performance with the heuristic approaches when combining three of the factors in the method *WR(RPOP,DR,0.5)*.

With the learning-based approaches DeepPredict and RFPredict we furthermore tried to consider all of these factors in a more elegant and automated way. And, in fact, applying a neural network in our DeepPredict approach in combination with the *C-KNN* method led to the overall best results in our experiments for all three datasets. The difference between the best-performing method and any other method was statistically significant at $p < 0.05$ according to a Student's t-test with Bonferroni correction.

The random forest based method RFPredict was in some test configurations slightly better than the fine-tuned weighted hybrid approach. In several other situations, we could however not find a parameter setting for the random forest classifier that led to an improvement over the heuristics-based hybrid. A possible reason for this observation might lie in the more coarse-grained predictions of the purchase probabilities of this method when compared with the neural network based method.

Overall, our results show that re-ranking the recommendations by considering a number of additional session-based factors can lead to significant accuracy gains. Furthermore, the experiments provide further evidence that modern deep learning approaches can help to obtain better results in this domain than we could achieve with more traditional methods like random forests or with manually tuned weight factors.

To further validate our observations, we have conducted additional experiments on different subsamples as described in Section 2. The datasets include a set of 3,000 *random* users who made at least one purchase, and two larger subsamples of *regular* users, involving 5,000 and 10,000 regular users, respectively. The results confirm the observations that were made in this section, i.e., that the best results can be obtained with the DeepPredict method, i.e., the effectiveness of the method is not limited to a certain type of users. The dataset characteristics and the detailed results are listed in the appendix (Tables 17 and 18).

4.6 Practical Implications

Our work has a number of practical implications. First, the work presented in this paper in general sheds light on the relative importance of considering long-term preference models and the short-term situational aspects in real-world e-commerce recommendation processes. Our results show that considering, for example, the user's short-term interests can lead to substantially more effective recommendations, i.e., to recommendations that ultimately lead to a purchase, than when considering only long-term models that are re-trained overnight. From a practical perspective it is interesting to see that already quite simple heuristic methods, which can be applied to re-rank the recommendation lists in real-time, are comparably effective. Furthermore, these heuristic methods can be used as a fallback method for anonymous or one-time users for which no long-term user profiles exist.

Second, our research works provide evidence that recommending items that the user has already seen can be a promising strategy in some domains, and we have proposed novel heuristics of how to select the items that the user should be reminded of. Practitioners should therefore consider and evaluate the inclusion of reminder items in their recommendations. In practical environments, the selection of the reminder items and how many of the items of a recommendation list should be reminders depends on the specifics of the domain and has to be determined based on business considerations or A/B tests. The analysis of the log data that was available to us indicates that at least some companies include a substantial amount of reminders in the recommendations.

Third, our analyses showed that customers of the analyzed shop click more often on recommendations when they relate to recently trending items or items that are currently discounted. Regarding the discounts, this means that users generally have no negative connotations when seeing recommendations that are on sale. They seemingly do not consider them as being less relevant, which might be the case for other advertisements on the shop website. Whether or not increased sales of already trending or discounted items are desirable in practice depends on the specifics of the business and the intended purpose of the recommendation service. If more revenue is the goal, selling more, even at reduced prices, is often better. If in contrast profit is the target business metric, promoting items with higher margins (including not-discounted ones), might be better. Similar considerations apply for the recommendation of already popular and trending items.

Fourth, our work shows that "reverse engineering" the most important features of successful recommendations from log data and using the features in a recommendation algorithm can be a promising approach in practice. Our work revealed some features that are probably relevant and useful in many e-commerce scenarios, like trends and discounts. In practice, often much more fine-grained information about the items or the contextual situation of the user is available, which can be used in the feature engineering phase. Considering a variety of such features in parallel can furthermore "automatically" result in diverse recommendation lists that represent a balanced mix of items that match

the user's long-term or short-term preferences, items the user has recently inspected, or items that sell well at the moment. The proposed prediction method is in general not limited to a certain set of features or specific learning method. In fact, our results showed that already the comparably simple weighted hybrid method can lead to good results. In practice, however, the optimal weights have to be fine-tuned based on offline experiments or through field tests in case there exists no suitable offline proxy for the target business metric. The proposed deep learning approach has a higher predictive power in offline experiments. In practice, however, one has to thoroughly analyze if the additional computational complexity of such a method justifies its usage.

5 Related Works

Historically, many papers in the field of recommender systems, like ours, target e-commerce domains, including the recommendation of books to purchase, movies to rent, or other types of goods to buy. Differently from many works in the literature, our work is not based on the matrix completion problem formulation, but addresses session-based recommendation scenarios, where the goal is to predict the users' next action(s) based on their most recent behavior. It therefore also falls into the category of *context-aware* recommender systems, where the context – in our case the user's short-term shopping intent – has to be inferred from the user's recent interactions with the website.

According to the categorization scheme for context-aware recommenders proposed by Adomavicius and Tuzhilin (2011), the techniques proposed in our work can be assigned to the category of *contextual post-filtering*. Post-filtering means that we initially create a list of items that are potentially relevant for a given user in general and then reorganize this list based on the currently given contextual situation.

Within the category of context-aware recommenders itself, our work falls into what can be called "session-based" or "session-aware" recommendation approaches. In session-based recommenders, the algorithm is typically given information about the last few interactions of the user with the system and the problem is to predict a certain future event, e.g., the next item view or purchase event, in the given session. "Next-basket recommendation", as discussed, e.g., in (Rendle et al, 2010), represents a specific form of this problem in e-commerce settings. Session-based next-item recommendations are however also common in the music domain in the form of playlist generation problems (Hariri et al, 2012; Bonnin and Jannach, 2014) or in the context of the provision of automated website navigation aids (Mobasher et al, 2002).

Compared to the huge amount of works on non-contextualized recommendations based on public datasets from MovieLens, Yahoo! or Netflix, works on session-based recommendation problems are comparably scarce. Early works in this area were published mostly in the field of *interactive* and *knowledge-based* recommender systems. In such approaches, the users are typically asked directly about their short-term preferences, e.g., using interactive queries or critiquing-based approaches (Ricci et al, 2003; Burke, 2000).

An example of an early *learning-based approach* for the problem of modeling short-term intents and predicting the next user action is the work by Mobasher et al (2002). Their goal was to predict the next navigation actions of users on a websites. Technically, sequential pattern mining was proposed as one solution to find patterns in past interaction logs. Later on, related works on the extraction of repeated navigation patterns from log data were proposed by Aghabozorgi and Wah (2009) and AlMurtadha et al (2010). The *C-CoOcc* method used in our experiments can be seen as a simple form of pattern mining. Sequential patterns for next-item recommendation were also evaluated for the music recommendation domain by Bonnin and Jannach (2014). In their work, it how-

ever turned out that sequential patterns were not as effective as a neighborhood-based method.

Session-based recommendations are also discussed in the domain of personalized news. Here, the recency of news items is crucial for successful recommendations, as the information gets irrelevant quickly. Therefore, focusing of the short-term interests of individual users or user groups becomes highly important. For example, in (Garcin et al, 2013) the authors create a news recommender with context trees that incrementally recommends news articles based on the current click stream of the users. The approach is therefore applicable for anonymous users that are not logged in and have no long-term profile. Similar to that work, Li et al (2010) utilize contextual bandits to generate recommendations for *Yahoo! News* based on the user's visited news article pages. For each page a user visits, the system sequentially sends new recommendations to the user and is therefore able to detect their short-term interests. A different news recommendation approach was proposed for the *Google News* platform by Liu et al (2010). The authors present a hybrid approach that uses content-based information about news items that match the user's interests, but also employs a collaborate filtering algorithm that detects the short-term trends based on the interactions of a user's neighborhood.

Another intuitive form of considering session-based next-item recommendation problems from a machine learning perspective is to view them as sequential optimization problems and to model them as Markov Decision Processes (Shani et al, 2005). Later papers that considered Markov processes or Markov chains for next-item recommendation problems include the works described in (Rendle et al, 2010) and (Tavakol and Brefeld, 2014) for the e-commerce and fashion domains or (Chen et al, 2012) for the music domain. In principle, these approaches can be used as alternative baseline techniques in our two-phase re-ranking scheme. In contrast to our approach, which considers several additional factors like reminders or recency aspects, the above-mentioned works mostly focus on determining those items that match the current intent of the session. In some cases, Markov process based approaches can suffer from scalability issues and evidence exists that in some domains neighborhood-based methods are at least equally effective in terms of their prediction accuracy (Bonnin and Jannach, 2014).

Recurrent neural networks (RNNs), as a special form of artificial neural networks, represent another machine-learning approach to model the next-item prediction problem. Such approaches recently gained popularity in the context of *deep learning* models. Examples of works that aim to learn the dynamic temporal behavior of users from log data with RNNs can be found, e.g., in (Romov and Sokolov, 2015) or (Hidasi et al, 2016). In contrast to our approaches, the mentioned works focus solely on short-term models, i.e., their predictions are only based on the actions of the current session. Again, these methods can in principle be used as alternatives baselines for our re-ranking models that are able to consider other aspects in the recommendation process. Recent work however revealed that despite the computational complexity of these models, they are not always favorable in terms of prediction accuracy over the *C-KNN* method used in our work (Jannach and Ludewig, 2017b). Another approach using RNNs for next-basket predictions was recently proposed in (Yu et al, 2016). Their method is capable of considering multiple sessions of a user over time, but only examines past checkout events and not a multitude of relevance signals as done in our approaches.

A number of alternative modeling techniques to deal with short-term and long-term interests was proposed in the literature. An early knowledge-based and "scenario-based" method was introduced in (Shen et al, 2007) to understand the user's immediate shopping needs in the fashion domain. Approaches to combine short-term interest models and long-term models were put forward, for example, by Anand and Mobasher (2007) or Nguyen and Ricci (2008). More recently, Hariri et al (2014) relied on a multi-arm bandit algorithm to consider the user's short-term goals and to discover possible interest shifts.

Finally, in the context of the ACM RecSys 2015 challenge, where the task was to predict whether or not a purchase will be made in a session and which item will be purchased, Romov and Sokolov (2015) used gradient boosting and decision trees to make predictions in a two-stage classification process. Similar to the DeepPredict method proposed in our work, Romov and Sokolov (2015) designed a number of features to predict the next item using a classification-based approach. In our work, we however consider additional types of features like reminders that were not in the focus of previous research. Moreover, the experiments on our datasets showed that artificial neural networks were able to outperform classifiers based on random forests.

Generally, considering *reminders* within recommendation lists has not been in the focus of the recommender systems research to a large extent. In an early work, Prassas et al (2001) proposed to remind users of online shopping sites of products during the "checkout" process ("Don't forget to buy"); Plate et al (2006) later on developed a mobile shopping assistant which suggests known items that are related to the products that the user is currently inspecting. Both mentioned articles are examples of works that discuss the potential value of reminding users of known things. However, no algorithmic approaches to select the items to recommend are proposed.

Recommending known items for *repeated consumption* was for example discussed by Anderson et al (2014) and Kapoor et al (2015). In the work of Anderson et al (2014), the recency of past interactions with a given product was considered as the best indicator for repeated consumption. Correspondingly, a recommendation model for known items was designed that incorporates the time of an item's last consumption with an exponential decay factor. Kapoor et al (2015) focus on music recommendations where repeated consumptions occur more often than in the online shopping domain. In their work the authors present a method that estimates, based on the current context, whether a user is in the mood for listening to new tracks or not. Overall, while the work of Anderson et al (2014) has similarities with our reminding strategies, we see the integration of the ideas of Kapoor et al (2015) as a potential future extension to our reminding approaches. Repeated purchases of the *exact* same item are however not very common in our dataset from the fashion domain and it is not fully clear yet if the findings of Kapoor et al (2015) that were made in the music domain generalize to this domain.

Reminders however may not only serve as a means to point users to items that were of interest in the past, they can also represent a form of *navigation shortcuts*, e.g., when users repeatedly inspect the different items of their current choice set. To which extent navigation shortcuts ("shortlists") are adopted by users was in the focus of a study by Schnabel et al (2016). Their work revealed that users in fact heavily relied on the functionality of a manual shortlist creation tool that was made available to them during the study. For cases in which such a functionality is not available, Close and Kukar-Kinney (2010) in an earlier work discovered that some online users tend to misuse the shopping basket for the purpose of managing the candidate products.

Finally, regarding the question of how to consider (short-term) popularity trends and whether or not to recommend discounted items, limited academic research exists so far. Recommending *generally popular* items is a common approach in cold-start situations, even though such recommendations often do not lead to the best results in terms of the business value (Adomavicius and Tuzhilin, 2005; Jannach and Hegelich, 2009). Considering items that are *currently popular* on the site was recently discussed by Padmanabhan et al (2015) and Gomez-Uribe and Hunt (2015) in the context of the movie and TV show recommendations provided by Netflix. Besides the consideration of recent trends, Gomez-Uribe and Hunt (2015) state that their algorithms generally rely on a "pretty healthy dose of (unpersonalized) popularity". In the academic environment, in contrast, the recommendation of popular items is often considered as being of limited value. More research is therefore required to understand in which domains and under

which circumstances the recommendation of known and generally popular items can contribute value to the business.

With respect to the recommendations of discounted items, to our knowledge very limited academic research has been published in the computer science literature until now. The topic of *dynamic* and *personalized pricing*, which is extensively discussed, e.g., by Choudhary et al (2005), is generally to some extent related. Werro et al (2005) for example propose a method for the generation of personalized discounts to retain promising customers and increase the willingness to purchase products. Personalized prices were also in the focus of the later work by Kamishima and Akaho (2011), who propose to dynamically adjust the prices in the context of a recommendation system. Research in this field is unfortunately hampered by the lack of real-world datasets, and works like the one by Kamishima and Akaho (2011) can currently only be based on synthetic datasets and simulations.

6 Summary, Research Limitations, and Future Works

The goal of our work was to contribute to a better understanding of factors that can make e-commerce recommendations successful in practice. Specifically, we could show that including already known, trending, and currently discounted items within the recommendations of an e-commerce site can be useful. These aspects, as well as the consideration of the user's short-term shopping intents, are so far little explored in the literature, partially due to the lack of publicly available datasets.

Our technical approach is based on the systematic analysis of characteristics of successful real-world recommendations, which to our knowledge has not been done in the e-commerce domain before. The obtained insights helped us engineer new methods that consider various signals in parallel and lead to higher prediction accuracy than previous methods. The relevance of specific features for the prediction task might be domain-specific, the general research approach is however generic and can be applied in other domains as well. We identified the features that we used in our analyses based on the research literature, based on observations from practical systems, and based on general considerations regarding the behavior of markets. Depending in particular on the application domain, certainly also other factors can exist that could be considered in addition to those discussed in our work.

So far, we could test our integrated deep learning model only in one application domain for which we had a dataset available that contained all relevant information including the pricing. Individual aspects of our approach were however already validated for different datasets and domains. The reminders were for example evaluated in a field test as described above. The importance of considering short-term intents was analyzed for a second e-commerce dataset (TMall) in (Jannach et al, 2015a). Using the TMall dataset, we could also validate that recommending items that were popular in the last few days is a better strategy than to recommend those items that were the most popular ones during the entire data collection period.

More research is however still required in different directions. First, while the general approach of deriving characteristics of successful recommendations from log data is applicable in domains other than e-commerce, it is not clear if the specific aspects investigated in this paper also play an important role in different scenarios. Recommending discounted items can for example be explored in other domains as well where the goal of the recommendations is to stimulate purchases. It is furthermore intuitive to assume that reminders and current trends should be factors to be considered for example in the music recommendation domain. However, determining a suitable moment to remind users of songs they heard in the past might be more challenging as short-term musi-

cal preferences can be influenced by a number of factors including the user's mood or current willingness to explore new things (Kapoor et al, 2015).

To address such issues, in general better methods are still needed to automatically assess the user's current intent and motivation to visit the site. Furthermore, a common challenge in practical applications also outside the e-commerce domain is to find the optimal balance between the recommendation of novel items, generally trending items, and reminders. Placing these different sets of items in separate recommendation lists is not uncommon on real-world e-commerce platforms. Barely any research on multiple recommendation list exists in the literature, even though this appears to be a problem that is relevant in practice (Gomez-Uribe and Hunt, 2015).

Finally, the question regarding the true business value of different recommendation strategies can probably only be answered in the context of a specific application. Our work shows at least for the case of reminders that they led not only to better results in the offline experiments, but also to a measurable increase in terms of the business metric on the e-commerce site on which we could run a field test.

References

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Transactions on Knowledge and Data Engineering 17(6):734–749

Adomavicius G, Tuzhilin A (2011) Context-aware recommender systems. In: Recommender Systems Handbook, Springer, pp 217–253

Aghabozorgi SR, Wah TY (2009) Recommender systems: Incremental clustering on web log data. In: Proceedings of the 2Nd International Conference on Interaction Sciences: Information Technology, Culture and Human, ICIS '09, pp 812–818

AlMurtadha Y, Sulaiman NB, Mustapha N, Udzir NI, Muda Z (2010) ARS: Web page recommendation system for anonymous users based on web usage mining. In: Proceedings of the European Conference of Systems, and European Conference of Circuits Technology and Devices, and European Conference of Communications, and European Conference on Computer Science, ECS'10/ECCTD'10/ECCOM'10/ECCS'10, pp 115–120

Anand S, Mobasher B (2007) Contextual recommendation. In: From Web to Social Web, Springer, pp 142–160

Anderson A, Kumar R, Tomkins A, Vassilvitskii S (2014) The dynamics of repeat consumption. In: Proceedings of the 23rd International Conference on World Wide Web, WWW '14, pp 419–430

Berry S, Levinsohn J, Pakes A (1995) Automobile prices in market equilibrium. Econometrica 63(4):841–890

Bonnin G, Jannach D (2014) Automated generation of music playlists: Survey and experiments. ACM Computing Surveys 47(2):26:1–26:35

Burke R (2000) Knowledge-based recommender systems. Encyclopedia of Library and Information Science 69(32):180–200

Candel A, Parmar V, LeDell E, Arora A (2017) Deep learning with H2O. `http://docs.h2o.ai/h2o/latest-stable/h2o-docs/booklets/DeepLearningBooklet.pdf`, accessed 17 August 2017

Chen S, Moore JL, Turnbull D, Joachims T (2012) Playlist prediction via metric embedding. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, pp 714–722

Choudhary V, Ghose A, Mukhopadhyay T, Rajan U (2005) Personalized pricing and quality differentiation. Management Science 51(7):1120–1130

Close AG, Kukar-Kinney M (2010) Beyond buying: Motivations behind consumers' online shopping cart use. Journal of Business Research: Advances in Internet Consumer Behavior & Marketing Strategy 63(9–10):986–992

Diehl K, van Herpen E, Lamberton C (2015) Organizing products with complements versus substitutes: Effects on store preferences as a function of effort and assortment perceptions. Journal of Retailing 91(1):1–18

Garcin F, Dimitrakakis C, Faltings B (2013) Personalized news recommendation with context trees. In: Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, pp 105–112

Garcin F, Faltings B, Donatsch O, Alazzawi A, Bruttin C, Huber A (2014) Offline and online evaluation of news recommender systems at swissinfo.ch. In: Proceedings of the 8th ACM Conference on Recommender Systems, RecSys '14, pp 169–176

Gomez-Uribe CA, Hunt N (2015) The Netflix recommender system: Algorithms, business value, and innovation. Transactions on Management Information Systems 6(4):13:1–13:19

Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press, `http://www.deeplearningbook.org`, accessed 17 August 2017

Hariri N, Mobasher B, Burke R (2012) Context-aware music recommendation based on latent topic sequential patterns. In: Proceedings of the Sixth ACM Conference on Recommender Systems, RecSys '12, pp 131–138

Hariri N, Mobasher B, Burke R (2014) Context adaptation in interactive recommender systems. In: Proceedings of the 8th ACM Conference on Recommender Systems, RecSys '14, pp 41–48

Hidasi B, Karatzoglou A, Baltrunas L, Tikk D (2016) Session-based recommendations with recurrent neural networks. In: Proceedings of the International Conference on Learning Representations, ICLR '16

Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov R (2012) Improving neural networks by preventing co-adaptation of feature detectors. CoRR abs/1207.0580

Jannach D, Adomavicius G (2016) Recommendations with a purpose. In: Proceedings of the 10th ACM Conference on Recommender Systems, RecSys '16, pp 7–10

Jannach D, Hegelich K (2009) A case study on the effectiveness of recommendations in the mobile internet. In: Proceedings of the 3rd ACM Conference on Recommender Systems, RecSys '09, pp 205–208

Jannach D, Ludewig M (2017a) Determining characteristics of successful recommendations from log data – a case study. In: Proceedings of the Symposium on Applied Computing, SAC '17, pp 1643–1648

Jannach D, Ludewig M (2017b) When recurrent neural networks meet the neighborhood for session-based recommendation. In: Proceedings of the 11th ACM Conference on Recommender Systems, RecSys '17, p (forthcoming)

Jannach D, Lerche L, Jugovac M (2015a) Adaptation and evaluation of recommendations for short-term shopping goals. In: Proceedings of the 9th ACM Conference on Recommender Systems, RecSys '15, pp 211–218

Jannach D, Lerche L, Kamehkhosch I (2015b) Beyond "hitting the hits" – generating coherent music playlist continuations with the right tracks. In: Proceedings of the 9th ACM Conference on Recommender Systems, RecSys '15, pp 187–194

Jannach D, Lerche L, Kamehkhosh I, Jugovac M (2015c) What recommenders recommend: an analysis of recommendation biases and possible countermeasures. User Modeling and User-Adapted Interaction 25(5):427–491

Kamishima T, Akaho S (2011) Personalized pricing recommender system: Multi-stage epsilon-greedy approach. In: Proceedings of the 2Nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems, HetRec '11, pp 57–64

Kapoor K, Kumar V, Terveen L, Konstan JA, Schrater P (2015) "I like to explore sometimes": Adapting to dynamic user novelty preferences. In: Proceedings of the 9th ACM Conference on Recommender Systems, RecSys '15, pp 19–26

Lerche L, Jannach D, Ludewig M (2016) On the value of reminders within e-commerce recommendations. In: Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, UMAP '16, pp 27–25

Li L, Chu W, Langford J, Schapire RE (2010) A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th International Conference on World Wide Web, WWW '10, pp 661–670

Liu J, Dolan P, Pedersen ER (2010) Personalized news recommendation based on click behavior. In: Proceedings of the 15th International Conference on Intelligent User Interfaces, IUI '10, pp 31–40

Manning CD, Raghavan P, Schütze H (2008) Introduction to Information Retrieval. Cambridge University Press, New York, NY, USA

Mobasher B, Dai H, Luo T, Nakagawa M (2002) Using sequential and non-sequential patterns in predictive web usage mining tasks. In: Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM '02, pp 669–672

Moe WW (2003) Buying, searching, or browsing: Differentiating between online shoppers using in-store navigational clickstream. Journal of Consumer Psychology 13(1):29–39

Nguyen QN, Ricci F (2008) Long-term and session-specific user preferences in a mobile recommender system. In: Proceedings of the 13th International Conference on Intelligent User Interfaces, IUI '08, pp 381–384

Padmanabhan P, Sadekar K, Krishnan G (2015) What's trending on Netflix? URL `https://medium.com/netflix-techblog/whats-trending-on-netflix-f00b4b037f61`, accessed 17 August 2017

Plate C, Basselin N, Kröner A, Schneider M, Baldes S, Dimitrova V, Jameson A (2006) Recomindation: New functions for augmented memories. In: Proceedings of the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, AH '06, pp 141–150

Prassas G, Pramataris KC, Papaemmanouil O, Doukidis GJ (2001) A recommender system for online shopping based on past customer behaviour. In: Proceedings of the 14th BLED Electronic Commerce Conference, BLED '01, pp 766–782

Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L (2009) BPR: Bayesian personalized ranking from implicit feedback. In: Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, UAI '09, pp 452–461

Rendle S, Freudenthaler C, Schmidt-Thieme L (2010) Factorizing personalized markov chains for next-basket recommendation. In: Proceedings of the 19th International Conference on World Wide Web, WWW '10, pp 811–820

Ricci F, Venturini A, Cavada D, Mirzadeh N, Blaas D, Nones M (2003) Product recommendation with interactive query management and twofold similarity. In: Proceedings of the 5th International Conference on Case-Based Reasoning, ICCBR '03, pp 479–493

Romov P, Sokolov E (2015) Recsys challenge 2015: Ensemble learning with categorical features. In: Proceedings of the 2015 International ACM Recommender Systems Challenge, RecSys '15 Challenge, pp 1:1–1:4

Schnabel T, Bennett PN, Dumais ST, Joachims T (2016) Using shortlists to support decision making and improve recommender system performance. In: Proceedings of the 25th International Conference on World Wide Web, WWW '16, pp 987–997

Shani G, Heckerman D, Brafman RI (2005) An MDP-Based Recommender System. Journal of Machine Learning Researh 6:1265–1295

Shen E, Lieberman H, Lam F (2007) What am I gonna wear?: Scenario-oriented recommendation. In: Proceedings of the 12th International Conference on Intelligent User Interfaces, IUI '07, pp 365–368

Sutskever I, Martens J, Dahl G, Hinton G (2013) On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on International Conference on Machine Learning, ICML '13, pp 1139–1147

Tavakol M, Brefeld U (2014) Factored MDPs for detecting topics of user sessions. In: Proceedings of the 8th ACM Conference on Recommender Systems, RecSys '14, pp 33–40

Wager S, Wang S, Liang P (2013) Dropout training as adaptive regularization. In: Proceedings of the 26th International Conference on Neural Information Processing Systems, NIPS'13, pp 351–359

Werro N, Stormer H, Meier A (2005) Personalized discount - a fuzzy logic approach. In: Proceedings of the 5th IFIP Conference on e-Commerce, e-Business, and e-Government, I3E '05, pp 375–387

Yu F, Liu Q, Wu S, Wang L, Tan T (2016) A dynamic recurrent model for next basket recommendation. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '16, pp 729–732

Zeiler MD (2012) ADADELTA: an adaptive learning rate method. CoRR abs/1212.5701

Author Biographies

Dr. Dietmar Jannach is a Professor of Computer Science at TU Dortmund, Germany and head of the department's e-services research group. Dr. Jannach has worked on different areas of artificial intelligence, including recommender systems, model-based diagnosis, and knowledge-based systems. He is the leading author of a textbook on recommender systems and has authored more than hundred technical papers, focusing on the application of artificial intelligence technology to practical problems.

Malte Ludewig is a PhD candidate in Computer Science at TU Dortmund, Germany, from where he also received his MSc degree. His research interests lie in the field of recommender systems – with a focus on session-based recommendations – and personalization in e-commerce environments in general.

Dr. Lukas Lerche obtained his PhD and MSc degrees from TU Dortmund, Germany. During his doctoral studies Dr. Lerche worked on different research problems in the field of recommender systems. In his research publications, he mostly focused on recommendations based on implicit feedback and on session-based recommendation scenarios in e-commerce.

Appendix

A Numbers of Purchases per User

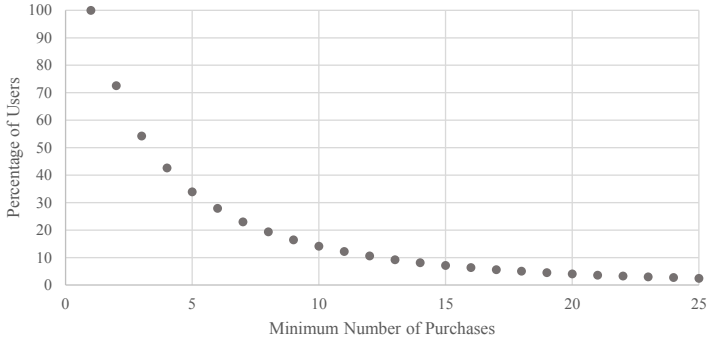


Fig. 6: The figure visualizes the purchase frequency distribution in the raw dataset, considering only users who ever made a purchase during the data collection period. The X-axis represents the minimum number of past purchases and on the Y-axis, the percentage of users in the dataset is shown that are above this threshold. For example, about one third of all users made 5 or more purchases.

B Feature Weights for the Occasional Users

Table 14: Results of the statistical feature weight analysis for the *occasional* user subset.

Gain Ratio analysis	
Feature	Weight
Discount level	0.453
Any discount granted?	0.325
Current popularity (day)	0.294
Current popularity (week)	0.231
Viewed before?	0.191
Distance to first view (in days)	0.191
Distance to first view (in sessions)	0.191
Current popularity (month)	0.174
Distance to last view (in sessions)	0.165
Distance to last view (in days)	0.157

Chi Squared analysis (normalized)	
Feature	Weight
Current popularity (day)	1.000
Current popularity (week)	0.788
Any discount granted?	0.762
Discount level	0.762
Current popularity (month)	0.544
Number of previous views	0.371
Distance to last view (in days)	0.370
Distance to last view (in sessions)	0.369
Viewed before?	0.364
Distance to first view (in days)	0.364

C Examined Features

Table 15: The full list of the examined features (see Section 2.3) along with their type and a short explanation.

Feature Name	Type	Explanation
clicked	Label	Recommended item was clicked
clicked_wished	Label	Recommended item was clicked and added to the wish list
clicked_cart	Label	Recommended item was clicked and added to the cart

clicked_bought *Successful Recommendation*	Label	Recommended item was clicked and bought in the same or the next session
relpop_{day,week,month} *Current popularity (day,week,month)*	Numerical	Popularity of the item on the same day, in the same week, or in the same month
samebrand	Bool	Same brand as the currently viewed item
brandratio_session{1,2,3}	Numerical	Ratio of the recommended brand regarding actions in the last 1, 2, or 3 sessions
brandpop	Numerical	Overall popularity of the brand
brandpop_{day,week,month}	Numerical	Popularity of the brand on the same day, in the same week, or in the same month
samecolor	Bool	Same color as the currently viewed item
colorratio_session{1,2,3}	Numerical	Ratio of the recommended color regarding actions in the last 1, 2, or 3 sessions
colorpop	Numerical	Overall popularity of the color
colorpop_{day,week,month}	Numerical	Popularity of the color on the same day, in the same week, or in the same month
samecat_{1,2,3,4}	Bool	Same category as the viewed item on breadcrumb navigation level 1, 2, 3, or 4
catratio_session{1,2,3}_{1,2,3,4}	Numerical	Ratio of the recommended category (breadcrumb navigation level 1, 2, 3, or 4) regarding actions in the last 1, 2, or 3 sessions
catpop_{1,2,3,4}	Numerical	Overall popularity of the category on breadcrumb navigation level 1, 2, 3, or 4
catpop_{day,week,month}_{1,2,3,4}	Numerical	Popularity of the category on the same day, in the same week, or in the same month for breadcrumb navigation level 1, 2, 3, or 4
sameprice	Bool	Same price level as the currently viewed item
priceratio_session{1,2,3}	Numerical	Ratio of the recommended price level regarding actions in the last 1, 2, or 3 sessions
similarity_viewed	Numerical	Ratio of features matched with the currently viewed item
similarity_session{1,2,3}	Numerical	Average ratio of features matched with items from the last 1, 2, or 3 sessions
neighbors_color	Numerical	Ratio of neighbor recommendations with the same color
neighbors_brand	Numerical	Ratio of neighbor recommendations with the same brand
neighbors_price	Numerical	Ratio of neighbor recommendations with the same price level
neighbors_category_{1,2,3,4}	Numerical	Ratio of neighbor recommendations with the same category (breadcrumb navigation level 1, 2, 3, or 4)
neighbors_distance	Numerical	Average ratio of item features matched with neighbor recommendations
prevrecclicks_sim	Numerical	Average ratio of item features matched with previously clicked recommended items in a session
prevrecclicks_color	Numerical	Average ratio of matching colors with previously clicked recommended items in a session
prevrecclicks_brand	Numerical	Average ratio of matching brands with previously clicked recommended items in a session
prevrecclicks_cat_{1,2,3,4}	Numerical	Average ratio of matching categories with previously clicked recommended items in a session (breadcrumb navigation level 1, 2, 3, or 4)
boughtbefore_sim	Numerical	Average ratio of item features matched with the last three previously bought items
boughtbefore_color	Numerical	Average ratio of matching colors with the last three previously bought items
boughtbefore_brand	Numerical	Average ratio of matching brands with the last three previously bought items
boughtbefore_cat_{1,2,3,4}	Numerical	Average ratio of matching categories with the last three previously bought items (breadcrumb navigation level 1, 2, 3, or 4)
discount *Any discount granted?*	Nominal	Knowledge about a discount: yes/no/unknown

discount_level *Discount level*	Numerical	Level of the discount (-1:unknown, 0:none, 1:low, 2:medium, 3:high)
viewed_before *Viewed before?*	Bool	Has the recommended item been viewed before
viewed_before_count *Number of previous views*	Numerical	Counter of previous item views
viewed_before_days_min *Distance to last view (in days)*	Numerical	Shortest distance to a previous item view event in days
viewed_before_days_max *Distance to first view (in days)*	Numerical	Longest distance to a previous item view event in days
viewed_before_sessions_min *Distance to last view (in sessions)*	Numerical	Shortest distance to a previous item view event in sessions
viewed_before_sessions_max *Distance to first view (in sessions)*	Numerical	Longest distance to a previous item view event in sessions
rec_before	Bool	Has the recommended item been recommended before
rec _before_count	Numerical	Counter of previous item recommendations
rec_before_days_min	Numerical	Shortest distance to a previous item recommendation event in days
rec _before_days_max	Numerical	Longest distance to a previous item recommendation event in days
rec _before_sessions_min	Numerical	Shortest distance to a previous item recommendation event in sessions
rec _before_sessions_max	Numerical	Longest distance to a previous item recommendation event in sessions
avg_colors_session{1,2,3}	Numerical	Average number of colors in the last 1, 2, or 3 sessions
avg_brands_session{1,2,3}	Numerical	Average number of brands in the last 1, 2, or 3 sessions
avg_price_session{1,2,3}	Numerical	Average number of price levels in the last 1, 2, or 3 sessions
avg_cat{1,2,3,4}_session{1,2,3}	Numerical	Average number of categories (breadcrumb navigation level 1, 2, 3, or 4) in the last 1, 2, or 3 sessions
user_avg_colors_session	Numerical	Session-wise average of different colors over all past user sessions
user_avg_brands_session	Numerical	Session-wise average of different brands over all past user sessions
user_avg_price_session	Numerical	Session-wise average of different price levels over all past user sessions
user_avg_cat{1,2,3,4}_session	Numerical	Session-wise average of different categories (breadcrumb navigation level 1, 2, 3, or 4) over all past user sessions
user_price	Numerical	Average price level of the user regarding all past actions
user_pricelevel_{view,buy}	Numerical	Average price level of the user regarding all past view or buy actions
user_discount	Numerical	Average discount level of the user regarding all past actions
user_pricereduction_{view,buy}	Numerical	Average discount level of the user regarding all past view or buy actions
user_viewedbefore_click	Numerical	Ratio of the user clicking on already known recommendations
user_viewedbefore_click_count	Numerical	Average number of previous item views before clicking on a recommendation
user_viewedbefore_success	Numerical	Ratio of the user clicking already known recommended items and buying them later on in the same session
user_viewedbefore_success_count	Numerical	Average number of previous item views before a sucessful recommendation (click and buy in the same session)

Table 16: List of features not considered by DeepPredict.

Feature name
brandpop
brandpop_day
brandpop_week
catpop
catpop_month
catpop_week
catpop_day

D Additional Experimental Results

Table 17: Characteristics of the additional Zalando datasets.

	Raw dataset	3k *Random* users	5k *Regular* users	10k *Regular* users
Users	3.5M	3,000	5,000	10,000
Items	460k	76k	150k	185k
Views	200M	358k	1.6M	3.2M
Purchases	3.9M	12k	67k	134k
Sessions	27.5M	41k	146k	293k
Sessions per user	7.79	13.77	29.22	29.29
Views per session	7.28	8.74	11.15	11.03
Purchases per session	0.14	0.29	0.45	0.46

Table 18: *Hit Rate@10* and *MRR@10* results for the additional subsets of *random* and *regular* Zalando users. Statistically significant differences (according to a Student's t-test with $p < 0.05$) between DeepPredict and the second-best method in the experiments are marked with a star.

Baseline	C-KNN					
Dataset	Zalando *random* 3k		Zalando *regular* 5k		Zalando *regular* 10k	
Metric@10	HR	MRR	HR	MRR	HR	MRR
No post-processing	0.341	0.187	0.392	0.178	0.430	0.189
FM	0.381	0.207	0.415	0.183	0.450	0.194
IRec-FM	0.458	0.299	0.484	0.198	0.499	0.204
DR-FM	0.403	0.258	0.461	0.196	0.476	0.222
RPOP-FM	0.458	0.270	0.491	0.228	0.497	0.217
RFPredict	0.458	0.294	0.497	0.281	0.490	0.271
WR(RPOP,DR,0.5)-FM	0.467	0.309	0.513	0.262	0.517	0.250
DeepPredict	**0.480***	**0.355***	**0.523**	**0.294***	**0.547***	**0.316***

When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation

Dietmar Jannach
TU Dortmund, Germany
dietmar.jannach@tu-dortmund.de

Malte Ludewig
TU Dortmund, Germany
malte.ludewig@tu-dortmund.de

ABSTRACT

Deep learning methods have led to substantial progress in various application fields of AI, and in recent years a number of proposals were made to improve recommender systems with artificial neural networks. For the problem of making session-based recommendations, i.e., for recommending the next item in an anonymous session, Hidasi et al. recently investigated the application of recurrent neural networks with Gated Recurrent Units (GRU4REC). Assessing the true effectiveness of such novel approaches based only on what is reported in the literature is however difficult when no standard evaluation protocols are applied and when the strength of the baselines used in the performance comparison is not clear. In this work we show based on a comprehensive empirical evaluation that a heuristics-based nearest neighbor (kNN) scheme for sessions outperforms GRU4REC in the large majority of the tested configurations and datasets. Neighborhood sampling and efficient in-memory data structures ensure the scalability of the kNN method. The best results in the end were often achieved when we combine the kNN approach with GRU4REC, which shows that RNNs can leverage sequential signals in the data that cannot be detected by the co-occurrence-based kNN method.

CCS CONCEPTS

•Information systems →Recommender systems;
•General and reference →Evaluation;

KEYWORDS

Session-Based Recommendation; Deep Learning; Nearest-Neighbors

ACM Reference format:
Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In *Proceedings of RecSys '17, Como, Italy, August 27-31, 2017,* 5 pages.
DOI: http://dx.doi.org/10.1145/3109859.3109872

1 INTRODUCTION

Deep learning approaches based on artificial neural networks have recently led to significant advances in different application fields of AI like object classification in images, speech recognition, or game playing. Their success in these fields has inspired researchers to explore the potential of adapting deep learning methods for recommendation-related problems. While there are limited works yet that show the advantages of *directly* applying deep learning methods for the rating prediction task [29, 35], artificial neural networks were, for example, used to vectorize content features from audio, video or textual item data [1, 5, 6, 10] and in particular for the problem of *session-based recommendation* [14, 15, 30, 38].

The algorithmic task in the latter scenario is to predict the next action of a user given the sequence of the actions in the current session. The problem setting, while not largely explored in the research literature, is highly relevant in practical settings. Session-based approaches are usually applied when the visitors of the site are anonymous (not logged-in) and no past interactions of the users are known. Furthermore, considering the last few user actions is also important for applications where already known users often revisit the site with a specific short-term intent [18].

In their recent work, Hidasi et al. investigated the use of recurrent neural networks (RNN) for session-based next-item recommendation [14]. RNNs are a natural choice for this problem and have been successfully explored for other sequence-based prediction problems in the past [4, 8, 9, 16]. Technically, the approach in [14] uses a customized RNN with Gated Recurrent Units (GRU). An experimental evaluation on two datasets indicated that their GRU4REC method significantly outperforms other methods, including an item-based k-nearest-neighbor (kNN) method, which was the strongest baseline in their experiments. Despite these positive results, some questions regarding the effectiveness of the GRU4REC method remain open. The experiments in [14] were, for example, only conducted on two specific datasets, used baselines whose strength cannot be easily judged, and were based on a proprietary evaluation protocol without cross-validation.

To better understand the true effectiveness of the proposed approach, we conducted a series of experiments on multiple datasets in which we benchmarked the GRU4REC method with an alternative session-based nearest neighbor method, which was identified as a strong baseline in previous works on session-based music and e-commerce recommendation problems [2, 12, 21]. To achieve higher accuracy and to scale for larger datasets, our session-based kNN method incorporates heuristics to sample suitable neighbors. Our results show that the proposed kNN method leads to the same accuracy results as the best configuration reported in [14] and outperforms GRU4REC in many other tested problem setups.[1] Combining GRU4REC with the kNN methods in a weighted hybrid approach finally often led to the best results, which indicates that RNN methods are indeed capable of capturing sequential patterns in the data that the kNN approach could not identify.

RecSys '17, Como, Italy

978-1-4503-4652-8/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3109859.3109872

[1] Recent works suggest that advanced nearest-neighbor models can lead to competitive performance also for common item-ranking tasks [34].

2 EXPERIMENT CONFIGURATIONS

2.1 Algorithms

2.1.1 GRU4REC. We used the GRU4REC implementation in Python that the authors of [14] share online.[2] The code is regularly updated by the authors and includes the implementation of the GRU4REC method, the code of their baseline algorithms, as well as the code for the evaluation procedure used in [14].

2.1.2 Session-based kNN. The kNN method takes the set of user actions in the current session, e.g., two view events for certain items, and then in a first step determines the k most similar past sessions in the training data. Then, given the current session s, the set of k nearest neighbors N_s, and a function $sim(s1, s2)$ that returns a similarity score for two sessions $s1$ and $s2$, the score of a recommendable item i is

$$score_{\text{KNN}}(i, s) = \Sigma_{n \in N_s} sim(s, n) \times 1_n(i) \quad (1)$$

where $1_n(i) = 1$ if n contains i and 0 otherwise, see also [2]. We tested different distance measures. The best results were achieved when comparing the sessions, which were encoded as binary vectors of the item space, using cosine similarity.

Determining the similarity of the current session with millions of past sessions after each user action cannot easily be accomplished under the time constraints of online recommendation. We therefore pre-process the training sessions and create an in-memory index data structure (cache) on startup. Specifically, for each item we create an index that points to the sessions in which the item appears. For each session, we furthermore have a pointer to its set of items. When recommendations for a session s are needed, we first determine the set of *possible* neighbors by creating the union of sessions in which the items of s are contained. This is a fast operation as it only involves a cache lookup and set operations. From this set of possible neighbors, we create a subsample of m sessions randomly or using a heuristic. In some experiments, we took the most recent sessions in case such information was available as focusing on recent trends has shown to be effective for recommendations in e-commerce [19]. From m we select the k nearest neighbors regarding the current session s. Again through lookup and set union operations, we create the set of recommendable items R that appear in one of the k sessions. We then compute the score for the items in R using Equation 1. The set operations, similarity computations, and the final predictions can be done very efficiently, as will be discussed later in Section 3.2.3.

2.1.3 Hybrid Approach. We tested switching, cascading, as well as weighted hybrids of the GRU4REC and the kNN method. A weighted combination led to the best results in our experiments, where in the most successful configurations the kNN score was assigned a slightly higher weight. All source code and the public datasets used in our experiments can be found online. [3]

2.2 Datasets and Evaluation Protocols

We performed experiments both on variants of the ACM RecSys 2015 Challenge dataset (*RSC15* and *RSCW*) as used in [14], on the public e-commerce dataset used in the TMall competition (*TMALL*), as well as on three different datasets containing music playlists from the platforms last.fm (*LFM*), artofthemix.org (*AOTM*), and 8tracks.com (*8T*). Music playlists are different in nature from e-commerce user logs in various ways. Nonetheless, they are designed to be consumed in a listening session and the tracks are often arranged in a specific sequence by their creators. With the experiments on these datasets our goal is to assess if recurrent neural networks can capture sequential patterns in the data which are not leveraged by the co-occurrence-based kNN approach. The basic dataset statistics are shown in Table 1, where *RSCW* and *TMALL* correspond to the average characteristics when applying a sliding-window protocol, as will be described below. In [14], Hidasi et al. use GRU4REC to predict the next item view events in a session. They incrementally add events to the sessions in the test set and report the average hit rate (*HR*) and the Mean Reciprocal Rank (*MRR*) at list length 20. They trained the GRU4REC on six months of data and used one single day for the evaluation.

Table 1: Dataset characteristics

	RSC15	RSCW	TMALL	LFM	AOTM	8T
Sessions	8M	4M	650K	120K	82K	520K
Avg. length	3.97	3.92	7.5	28.24	11.48	9.20
Items	37K	34K	300K	200K	54K	200K

In addition to this procedure, we performed experiments where we created multiple train-test splits, consisting of 3 and 1 month of training data for the *RSC15* and TMall dataset, respectively, and the subsequent day as the test data.[4] This sliding-windows approach allows us to minimize the risk that the obtained results are specific to the single train-test split used in [14]. Furthermore, we used the algorithms to predict purchase events in the sessions, which was one of the original tasks of the ACM RecSys 2015 challenge. To investigate the performance of the methods at different stages of a session, we also measured the accuracy when predicting the *second* and the *last* view of each session. Finally, we varied the amount of the training data to see how the algorithms compare in case of more sparse data situations.

3 RESULTS

3.1 Accuracy Results

3.1.1 ACM RecSys 2015 Challenge Dataset. Table 2 shows the results when using the experimental configuration used in [14], ordered by the values for HR@20 as done in the original paper. We could reproduce their best hit rate and MRR results (using their optimal parameters) for the methods GRU4REC(1000,BPR) and GRU4REC(1000,TOP1), which use 1000 hidden units and the TOP1 and BPR's pairwise ranking loss function, respectively. In Table 2, we additionally include the results for list length ten, which might be more important in different application domains.

The method KNN_{MR}(500,1000), which uses the 500 nearest neighbors from the 1000 most recent candidate sessions, outperforms GRU4REC in all measurements except for the MRR in the TOP1

[2] https://github.com/hidasib/GRU4Rec
[3] http://bit.ly/2nfNldD
[4] Using larger training splits for the TMall dataset, due to its large number of items, led to prohibitively high computational costs by the GRU4REC method, which is why we report the results when using one month as training data for this dataset.

Table 2: Results when using the evaluation scheme of [14].

Method	HR@10	MRR@10	HR@20	MRR@20
WH(KNN,GRU,0.6,0.4)	**0.568**	0.256	**0.691**	0.265
WH(KNN,GRU,0.1,0.9)	**0.568**	**0.269**	0.666	**0.276**
KNN_{MR}(500,1000)	0.521	0.242	0.641	0.250
GRU4REC(1000,BPR)	0.517	0.235	0.636	0.243
GRU4REC(1000,TOP1)	0.517	0.261	0.623	0.268
KNN_{RAND}(500,1000)	0.499	0.235	0.616	0.242
GRU4REC(100,TOP1)	0.481	0.221	0.595	0.230

configuration. Combining the kNN-method with GRU4REC in a weighted approach (WH) leads to the best results. As in [14], the "winner" at list length 20 depends on the metric. Also, different weights for the hybrid method have to be applied to achieve the best results for a given list length.[5] The increases of the best configuration at length 20 are about 8% for the hit rate. A smaller increase was observed for the MRR when compared to the results of [14]. Even with a random sampling of candidate sessions (KNN_{RAND}(500,1000)) the kNN-method does not fall far behind and is consistently better than GRU4REC with 100 hidden units.

Table 3 shows the best results obtained with KNN_{MR}(500,1000) and GRU4REC(1000,TOP1) when using alternative measurements, including the sliding window protocol (*RSCW*), the prediction of the second (*SECOND*) and last item view (*LAST*), and the accuracy when predicting which item is purchased in a session (*BUYS*). In all experiments, the kNN method outperforms GRU4REC. For the sliding windows protocol, we applied the Wilcoxon signed-rank test over the five experiment runs, which revealed that the differences are statistically significant in terms of the hit rate ($\alpha = 0.05$).

3.1.2 Results for Other Datasets. Table 4 shows the results for the additional datasets. We again report the results at list length 20 for GRU4REC(1000,TOP1) and KNN(500,1000). For the *TMALL* dataset, we applied the sliding-window approach as for *RSCW*. Note that in this dataset each session is defined as the sequence of actions of a user during one day (resulting in a larger average session length as shown in Table 1). As the playlists have no timestamp attached, we randomly assigned each playlist to one of 31 buckets (days of the month) and used one of them as test data.

In the majority of the configurations, the kNN method outperformed GRU4REC both in terms of the hit rate and the MRR, in many cases strongly. Only on the last.fm dataset, GRU4REC worked better than the kNN approach; on the AOTM dataset, GRU4REC was furthermore better in terms of the MRR. Generally, however, the results obtained for the additional datasets indicate that the good performance of the kNN method on the *RSC15* dataset is not due to the specific nature of this dataset or the application domain.

3.2 Additional Analyses

3.2.1 Focusing on Recent Data. The *RSC15* dataset contains about 45,000 sessions per day. Since our kNN method only uses, e.g., one thousand *possible* neighbor sessions, which are not necessarily all from the same day, we were interested to what extent we can omit past sessions without compromising the accuracy results.

[5] In all experiments we tuned the parameters for the different algorithms using grid search. We optimized the hit rate on validation sets (subsets of the training sets). Due to the run time GRU4REC was only optimized with 100 layers as also done in [14].

Table 3: Additional measurements for the RSC15 dataset.

Dataset	RSCW		SECOND		LAST		BUYS	
M@20	HR	MRR	HR	MRR	HR	MRR	HR	MRR
KNN	0.621	0.267	0.716	0.355	0.446	0.196	0.758	0.290
GRU4REC	0.587	0.261	0.655	0.300	0.388	0.174	0.542	0.215

Table 4: Results for other datasets.

Dataset	TMALL		LFM		AOTM		8T	
M@20	HR	MRR	HR	MRR	HR	MRR	HR	MRR
KNN	0.370	0.171	0.078	0.011	0.068	0.008	0.050	0.010
GRU4REC	0.223	0.117	0.121	0.053	0.035	0.012	0.019	0.008

Figure 1 shows MRR results when we incrementally remove data from the training set, beginning with the oldest sessions. The values for the hit rate are similar. For the GRU4REC method, it seems sufficient to focus on the last 60 days. For the kNN method in contrast, it is sufficient to retain the information of the last two days. While this might come surprising, focusing on the most recent events has shown to be effective in the past in the domains of e-commerce and news recommendations [19, 23].

As mentioned above, we made additional experiments on the *RSC15* data and repeatedly sampled 1000 random neighbor sessions instead of the most recent ones. The average results were shown in the last row of Table 2. Using random neighborhood sampling leads to slightly lower accuracy results, which are however still higher than the ones obtained by GRU4REC with 100 hidden units.

3.2.2 Popularity Bias and Catalog Coverage. To assess possible recommendation biases, we measured the average popularity of the recommendations. Using a normalized popularity score for the top-20 recommendations (in the setup of [14]) we found that the kNN method tends to recommends slightly more popular items on average than GRU4REC (0.036 vs. 0.028). With respect to catalog coverage, we observed that the top-20 recommendations of GRU4REC include about 47% of the items at least once, which is slightly more than the kNN method covers (41%). This latter observation is not surprising, given the focus of the kNN method on

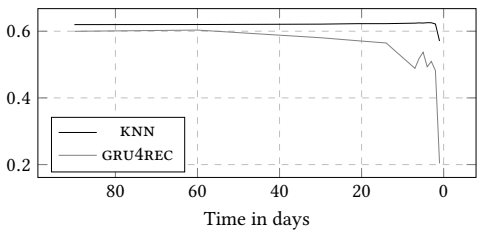


Figure 1: MRR@20 for the RSC15 when artificially reducing the size of the training set from 3 month to 1 day.

the last few days. Alternative neighbor sampling strategies can however be designed to deal, e.g., with such accuracy-coverage trade-offs.

3.2.3 Computational Complexity and Memory Usage. On a desktop computer with an Intel i7-4790k processor, training GRU4REC in the best configuration needed about 23 hours for the *RSC15* dataset, which can be reduced to about 8 hours when calculations are performed by the GPU (Nvidia GeForce GTX 960). The kNN method needs about 90 seconds to build the in-memory data structures. Creating one recommendation list with GRU4REC needed about 12 ms on average and 26 ms for the kNN method. Overall, computing all (about 40,000) recommendations needs about 27 minutes with the kNN method including data structure initialization, whereas GRU4REC needs more than 8 hours in total. At the same time, the kNN method has the advantage of supporting online updates. Further speed-ups for the kNN approach can in theory be achieved when the similarity computations are parallelized, since these calculations can be done independently for individual subsets of the neighbor candidates.

The data structures of the kNN method occupy about 6.4 GB of main memory when the entire *RSC15* training dataset (2.3 GB of raw data) is used. No special data structures are used in our implementation and given the observations from above, keeping only a specific amount of the most recent log actions will help to reduce the memory requirements. For the same dataset, GRU4REC's model needs about 60 MB for 100 and 600 MB for 1000 hidden units. GRU4REC's memory demand is thus dependent of the algorithm parameters and significantly increases with the number of items. In the playlist and TMall experiments, GRU4REC's memory demands exceeded the capacity of our graphic card, making computations very slow, which is why we could only make a limited number of evaluation runs on these datasets.

4 RELATED WORK

The commonly used algorithmic approaches to leverage sequential information for predicting next user actions include Markov models and sequential pattern mining techniques. In one of the earlier works on this topic, Shani et al., for example, proposed the application of Markov Decision Processes in the scenario of an online book store [27]. More recent works that rely in Markov models include [3, 13, 17, 26] or [31]. In [31], Markov models were, for example, used with the goal to detect topic sequences in user sessions for the next-item prediction task. Detecting and leveraging sequences of topics was also the goal in [11] for the next-track music recommendation scenario. In this case, however, the authors applied sequential pattern mining techniques. Sequential patterns have been investigated earlier also for the problem of predicting the online navigation behavior of users, e.g., in [25].

Approaches based on Markov models and sequential pattern mining represent alternative baselines for session-based recommendations. Not all approaches based on Markov model however scale too well for large datasets [27]. Sequential pattern mining approaches (and association rule mining approaches in general) often require some effort to find suitable thresholds for rule learning and, depending on the application domain, do not lead to better accuracy results than co-occurrence-based kNN methods [2].

Particularly in recent years, RNNs have been applied in different ways to leverage sequence information when recommending, often based on ideas of the LSTM model proposed in [16] to avoid the vanishing or exploding gradient problem.

Zhang et al. [38] for example successfully applied RNNs in a related setting of predicting sequential advertisement clicks, in which the partitioning into sessions is less important. Hidasi et al., who proposed GRU4REC [14], were among the first to explore RNNs for session-based recommendations. Using the same evaluation protocol and *RSC15* data set, Tan et al. in [30] proposed an enhanced version of GRU4REC, which is trained on embedded item sequences with random drop-out to reduce overfitting. The models are also retrained on the most recent sessions to better adapt to short-term trends. In their paper, the authors report significant improvements over [14] which would also slightly outperform our hybrid approach. The work confirms that considering recent trends is helpful for this *RSC15* dataset and domain. Experiments on other datasets were unfortunately not reported. Since no source code was available, we re-implemented their method based on the information in the paper, but could not reproduce their results.

In [15] and [32] RNN-based approaches were recently proposed that leverage additional item features to achieve higher accuracy. Including additional information about users and items in the recommendation process is also possible for kNN based methods, but was not in the focus of our work.

Combining short-term with long-term models has shown to be a successful strategy in the past, e.g., in the e-commerce domain [18]. Song et al. [28] recently proposed an approach to predicting news click sequences, which used an RNN to model short-term interests in a combination with long-term models. The kNN method used in this paper can be combined with long-term models as well. However, in the case of the *RSC15* dataset, where sessions have no user ID attached, no long-term user models can be learned.

Yu et al. [37] use RNNs for the related task of next basket recommendations, in which, e.g., items for a shopping cart are suggested based on a user's history of past shopping carts. Although the scenario is slightly different from the one in this paper, the kNN method can be easily adapted as a baseline for this scenario.

In general, NNs have been used for a number of recommendation-related tasks in recent years. Often, such networks are used to learn embeddings of content features in compact fixed-size latent vectors, e.g., for music, for images, for video data, for documents, or to represent the user [1, 5–7, 10, 20, 24, 36]. These representations are then integrated, e.g., in content-based approaches, in variations of latent factor models, or are part of new methods for computing recommendations [6, 7, 10, 22, 29, 33, 35].

5 CONCLUSIONS

Our work shows that nearest-neighbor methods should be considered as competitive baselines for session-based recommendation scenarios. Considering a mix of co-occurrence signals and sequential patterns, as identified by recurrent neural networks, led to the best results in our experiments. In practice, however, one has to decide based on the application domain if the obtained accuracy gains justify the usage of more complex learning methods. Our future works include the comparison of the performance of additional session-based algorithms like [26] or [30].

REFERENCES

[1] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-task Learning for Deep Text Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 107–114. DOI: http://dx.doi.org/10.1145/2959100.2959180

[2] Geoffray Bonnin and Dietmar Jannach. 2014. Automated Generation of Music Playlists: Survey and Experiments. *Computing Surveys* 47, 2 (Nov. 2014), 26:1–26:35. DOI: http://dx.doi.org/10.1145/2652481

[3] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist Prediction via Metric Embedding. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, 714–722. DOI: http://dx.doi.org/10.1145/2339530.2339643

[4] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014). http://arxiv.org/abs/1412.3555

[5] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 191–198. DOI: http://dx.doi.org/10.1145/2959100.2959190

[6] Sander Dieleman. 2016. Deep learning for audio-based music recommendation. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS '16)*. ACM, 1–1. DOI: http://dx.doi.org/10.1145/2988450.2991128

[7] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15)*. ACM, 278–288. DOI: http://dx.doi.org/10.1145/2736277.2741667

[8] Jeffrey L. Elman. 1990. Finding structure in time. *Cognitive Science* 14, 2 (1990), 179 – 211. DOI: http://dx.doi.org/10.1016/0364-0213(90)90002-E

[9] Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *CoRR* abs/1308.0850 (2013). http://arxiv.org/abs/1308.0850

[10] Yupeng Gu, Bo Zhao, David Hardtke, and Yizhou Sun. 2016. Learning Global Term Weights for Content-based Recommender Systems. In *Proceedings of the 25th International Conference on World Wide Web (WWW '16)*. ACM, 391–400. DOI: http://dx.doi.org/10.1145/2872427.2883069

[11] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware Music Recommendation Based on Latenttopic Sequential Patterns. In *Proceedings of the Sixth ACM Conference on Recommender Systems (RecSys '12)*. ACM, 131–138. DOI: http://dx.doi.org/10.1145/2365952.2365979

[12] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2015. Adapting to User Preference Changes in Interactive Recommendation. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI '15)*. AAAI, 4268–4274. http://dl.acm.org/citation.cfm?id=2832747.2832852

[13] Qi He, Daxin Jiang, Zhen Liao, Steven C. H. Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. 2009. Web Query Recommendation via Sequential Query Prediction. In *Proceedings of the 2009 IEEE International Conference on Data Engineering (ICDE '09)*. IEEE, 1443–1454. DOI: http://dx.doi.org/10.1109/ICDE.2009.71

[14] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *Proceedings of the International Conference on Learning Representations (ICLR '16)*. ACM. http://arxiv.org/abs/1511.06939

[15] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 241–248. DOI: http://dx.doi.org/10.1145/2959100.2959167

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (Nov. 1997), 1735–1780. DOI: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[17] Mehdi Hosseinzadeh Aghdam, Negar Hariri, Bamshad Mobasher, and Robin Burke. 2015. Adapting Recommendations to Contextual Changes Using Hierarchical Hidden Markov Models. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15)*. ACM, 241–244. DOI: http://dx.doi.org/10.1145/2792838.2799684

[18] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. 2015. Adaptation and Evaluation of Recommendations for Short-term Shopping Goals. In *Proceedings of the 9th ACM Conference on Recommender Systems (RecSys '15)*. ACM, 211–218. DOI: http://dx.doi.org/10.1145/2792838.2800176

[19] Dietmar Jannach and Malte Ludewig. 2017. Determining Characteristics of Successful Recommendations from Log Data – A Case Study. In *Proceedings of the 32th Annual ACM Symposium on Applied Computing (SAC '17)*. ACM.

[20] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 233–240. DOI: http://dx.doi.org/10.1145/2959100.2959165

[21] Lukas Lerche, Dietmar Jannach, and Malte Ludewig. 2016. On the Value of Reminders Within E-Commerce Recommendations. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization (UMAP '16)*. ACM, 27–35. DOI: http://dx.doi.org/10.1145/2930238.2930244

[22] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management (CIKM '15)*. ACM, 811–820. DOI: http://dx.doi.org/10.1145/2806416.2806527

[23] Cornelius A. Ludmann and H.-Jürgen Appelrath. 2016. Lessons Learned from Using a Data Stream Management System for Real-time Recommendation of Popular News Articles based on Real User Streams. In *Proceedings of the 4th Workshop on Large-Scale Recommender Systems at ACM RecSys 2016 (LSRS '16)*. ACM.

[24] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. ACM, 43–52. DOI: http://dx.doi.org/10.1145/2766462.2767755

[25] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. 2002. Using Sequential and Non-Sequential Patterns in Predictive Web Usage Mining Tasks. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM '02)*. IEEE, 669–672. DOI: http://dx.doi.org/10.1109/ICDM.2002.1184025

[26] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, 811–820. DOI: http://dx.doi.org/10.1145/1772690.1772773

[27] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *The Journal of Machine Learning Research* 6 (Dec. 2005), 1265–1295. http://dl.acm.org/citation.cfm?id=1046920.1088715

[28] Yang Song, Ali Mamdouh Elkahky, and Xiaodong He. 2016. Multi-Rate Deep Learning for Temporal Recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM Press, 909–912. DOI: http://dx.doi.org/10.1145/2911451.2914726

[29] Florian Strub, Romaric Gaudel, and Jérémie Mary. 2016. Hybrid Recommender System based on Autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS '16)*. ACM, 11–16. DOI: http://dx.doi.org/10.1145/2988450.2988456

[30] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS '16)*. ACM, 17–22. DOI: http://dx.doi.org/10.1145/2988450.2988452

[31] Maryam Tavakol and Ulf Brefeld. 2014. Factored MDPs for Detecting Topics of User Sessions. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, 33–40. DOI: http://dx.doi.org/10.1145/2645710.2645739

[32] Bartłomiej Twardowski. 2016. Modelling Contextual Information in Session-Aware Recommender Systems with Neural Networks. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 273–276. DOI: http://dx.doi.org/10.1145/2959100.2959162

[33] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, 225–232. DOI: http://dx.doi.org/10.1145/2959100.2959160 arXiv:1607.07326

[34] Koen Verstrepen and Bart Goethals. 2014. Unifying Nearest Neighbors Collaborative Filtering. In *Proceedings of the 8th ACM Conference on Recommender Systems (RecSys '14)*. ACM, New York, NY, USA, 177–184. DOI: http://dx.doi.org/10.1145/2645710.2645731

[35] Jeroen B. P. Vuurens, Martha Larson, and Arjen P. de Vries. 2016. Exploring Deep Space: Learning Personalized Ranking in a Semantic Space. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS '16)*. ACM, 23–28. DOI: http://dx.doi.org/10.1145/2988450.2988457

[36] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, 1235–1244. DOI: http://dx.doi.org/10.1145/2783258.2783273 arXiv:1409.2944

[37] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A Dynamic Recurrent Model for Next Basket Recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval (SIGIR '16)*. ACM, 729–732. DOI: http://dx.doi.org/10.1145/2911451.2914683

[38] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI '14)*. AAAI, 1369–1375. http://dl.acm.org/citation.cfm?id=2893873.2894086

A Comparison of Frequent Pattern Techniques and a Deep Learning Method for Session-Based Recommendation

Iman Kamehkhosh
TU Dortmund
iman.kamehkhosh@tu-dortmund.de

Dietmar Jannach
TU Dortmund
dietmar.jannach@tu-dortmund.de

Malte Ludewig
TU Dortmund
malte.ludewig@tu-dortmund.de

ABSTRACT

Making session-based recommendations, i.e., recommending items solely based on the users' last interactions without having access to their long-term preference profiles, is a challenging problem in various application fields of recommender systems. Using a coarse classification scheme, the proposed algorithmic approaches to this problem in the research literature can be categorized into *frequent pattern mining* algorithms and approaches that are based on *sequence modeling*. In the context of methods of the latter class, recent works suggest the application of recurrent neural networks (RNN) for the problem. However, the lack of established algorithmic baselines for session-based recommendation problems makes the assessment of such novel approaches difficult.

In this work, we therefore compare a state-of-the-art RNN-based approach with a number of (heuristics-based) frequent pattern mining methods both with respect to the accuracy of their recommendations and with respect to their computational complexity. The results obtained for a variety of different datasets show that in every single case a comparably simple frequent pattern method can be found that outperforms the recent RNN-based method. At the same time, the proposed much more simple methods are also computationally less expensive and can be applied within the narrow time constraints of online recommendation.

CCS CONCEPTS

•General and reference →Evaluation; •Information systems →Recommender systems; •Computing methodologies →*Neural networks; Rule learning;*

KEYWORDS

Session-Based Recommendations; Deep Learning; Frequent Pattern Mining; Benchmarking

1 INTRODUCTION

Making recommendations solely based on a user's current session and most recent interactions is a nontrivial problem for recommender systems. On an e-commerce website, for instance, when a visitor is new (or not logged in), there are no long-term user models that can be applied to determine suitable recommendations for this user. Furthermore, recent work shows that considering the user's short-term intent has often more effect on the accuracy of the recommendations than the choice of the method used to build the long-term user profiles [20]. In general, such types of problems are common on e-commerce sites, e.g., when returning users do not log in every time they use the site. The same challenges can, however, be observed also for other application domains, in particular for news and media (music and video) recommendation [21, 33].

The problem of predicting the next actions of users based solely on their sequence of actions in the current session is referred to in the literature as *session-based recommendation.* A number of algorithmic approaches have been proposed over the years to deal with the problem. Early academic approaches, for example, rely on the detection of sequential patterns in the session data of a larger user community. In principle, even simpler methods can be applied. Amazon's "Customers who bought …also bought" feature represents an example that relies on simple co-occurrence patterns to generate recommendations, in that case in the context of the very last user interaction (an item view event). A number of later works then explored the use of Markov models [30, 35, 39], and most recently, researchers explored the use of recurrent neural networks (RNN) for the session-based next-item recommendation problem [16, 17, 38, 42].

Today, RNNs can be considered one of the state-of-the-art methods for sequence learning tasks. They have been successfully explored for various sequence-based prediction problems in the past [5, 9, 11, 18] and in a recent work, Hidasi et al. [16] investigated an RNN variant based on gated recurrent units (GRU) for the session-based recommendations problem. In their work, they benchmarked their RNN-based method GRU4REC with different baseline methods on two datasets. Their results showed that GRU4REC is able to outperform the baseline approaches in terms of accuracy for top-20 recommendation lists.

While these results indicate that RNNs can be successfully applied for the given recommendation task, we argue that the experimental evaluation in [16] does not fully inform us about different aspects of the effectiveness and the practicability of the proposed method. First, regarding the effectiveness, it is unclear if the methods to which GRU4REC was compared are competitive. Second, as the evaluation was based on one single training-test split and only using accuracy measures, further investigations are necessary to assess, for example, if some algorithms exhibit certain biases, e.g., to recommend mostly popular items. Third, even if the RNN method is effective, questions regarding the scalability of the method should be discussed, in particular as hyper-parameter optimization for the complex networks can become very challenging in practice.

The goal of this work is to shed light on these questions and in the remainder of this paper we will report the detailed results of comparing a state-of-the-art RNN-based method with a number of computationally more efficient pattern mining approaches in different dimensions.

Workshop on Temporal Reasoning in Recommender Systems, collocated with ACM RecSys'17, Como, Italy.

2 PREVIOUS WORKS

In session-based recommendation problems, we are given a sequence of the most recent actions of a user and the goal is to find items that are relevant in the context of the user's specific short-term intent. One traditional way to determine recommendations given a set of recent items of interest is to apply frequent pattern mining techniques, e.g., based on association rules (AR) [1]. AR are often applied for market basket analysis with the goal to find sets of items that are bought together with some probability [14]. The order of the items or actions in a session is irrelevant for AR-based approaches. Sequential patterns mining (SP) [2] techniques, in contrast, consider the order of the elements in sessions when identifying frequent patterns. In one of the earlier works, Mobasher et al. [32] used frequent pattern mining methods to predict a user's next navigation action. In another work, Yap et al. [47] propose a sequential pattern-mining-based next-item recommendation framework, which weights the patterns according to their estimated relevance for the individual user. In the domain of music recommendation, Hariri et al. more recently [15] propose to mine sequential patterns of latent topics based on the tags attached to the tracks to predict the context of the next song.

A different way of finding item-to-item correlations is to look for sessions that are similar to the current one (*neighbors*), and to determine frequent item co-occurrence patterns that can be used in the prediction phase. Such neighborhood-based approaches were for example applied in the domains of e-commerce and music in [4] or [26]. In some cases and application domains, simple co-occurrence patterns are despite their simplicity quite effective, see, e.g., [20, 40] or [44].

Differently from such pattern- and co-occurrence-based techniques, a number of recent approaches are based on *sequence modeling* using, e.g., Markov models. The main assumption of Markov-model-based approaches in the context of session-based recommendation is that the selection of the next item in a session is dependent on a limited number of previous actions. Shani et al. [35] were among the first who applied first-order Markov chains (MC) for session-based recommendation and showed the superiority of sequential models over non-sequential ones. In the music domain, McFee and Lanckriet [30] proposed a music playlist generation algorithm based on MCs that – given a seed song – selects the next track from uniform and weighted distributions as well as from k-nearest neighbor graphs. Generally, a main issue when applying Markov chains in session-based recommendation is that the state space quickly becomes unmanageable when all possible sequences of user selections should be considered [12, 16].

More recent approaches to sequence modeling for session-based recommendation utilize recurrent neural networks (RNN). RNNs process sequential data one element at a time and are able to selectively pass information across sequence steps [28]. Zhang et al. [49], for example, successfully applied RNNs to predict advertisement clicks based on the users' browsing behavior in a sponsored search scenario. For session-based recommendations, Hidasi et al. [16] investigated a customized RNN variant based on gated recurrent units (GRU) [5] to model the users' transactions within sessions. They also tested several ranking loss functions in their solutions. Later on, in [17] and [42] RNN-based approaches were proposed which leverage additional item features to achieve higher accuracy. For the problem of news recommendation, Song et al. [36] proposed a temporal deep semantic structured model for the combination of long-term static and short-term temporal user preferences. They considered different levels of granularity in their model to process both fast and slow temporal changes in the users' preferences. In general, neural networks have been used for a number of recommendation-related tasks in recent years. Often, such networks are used to learn embeddings of content features in compact fixed-size latent vectors, e.g., for music, for images, for video data, for documents, or to represent the user [3, 6–8, 13, 25, 29, 46]. These representations are then integrated, e.g., in content-based approaches, in variations of latent factor models, or are part of new methods for computing recommendations [7, 8, 13, 27, 37, 43, 45].

In the work presented in this paper, we will compare different existing and novel pattern-mining-based approaches with a state-of-the-art RNN-based algorithm.

3 EXPERIMENT CONFIGURATIONS

3.1 Algorithms

3.1.1 RNN Baseline. GRU4REC is an RNN-based algorithm that uses Gated Recurrent Units to deal with the vanishing or exploding gradient problem proposed in [16]. In our experiments, we used the Python implementation that is shared by the authors online.[1]

3.1.2 Session-based kNN – KNN. The KNN method searches the k most similar past sessions ("neighbors") in the training data based on the set of items in the current session. Since the process of determining the neighbor sessions becomes very time-consuming as the number of sessions increases, we use an special in-memory index data structure (cache) in our implementation. Technically, in the training phase, we create a data structure that maps the training sessions to their set of items and one structure that maps the items to the sessions in which they appear. To make recommendations for the current session s, we first create a union of the sessions in which the items of s appear. This union will be the set of *possible* neighbors of the current session. This is a fast operation as it only involves a cache lookup and set operations. To further reduce the computational complexity of the prediction process, we select a subsample of these possible neighbors using a heuristic. In this work, we took the m most *recent* sessions as focusing on recent trends has shown to be effective for recommendations in e-commerce [23]. We then compute the similarity of these m most recent possible neighbors and the current session and select the k most similar sessions as the neighbor sessions of the current session. Again through lookup and set union operations, we create the set of *recommendable* items R that contains items that appear in one of the k sessions. For each recommendable item i in R, we then compute the KNN score as the sum of the similarity values of s and its neighbor sessions $n \in N_s$ which contains i (Equation 1). The indicator function $1_n(i)$ returns 1 if n contains i and 0 otherwise, see also [4].

$$score_{\text{KNN}}(i, s) = \Sigma_{n \in N_s} sim(s, n) \times 1_n(i) \quad (1)$$

In our experiments, we tested different distance measures to determine the similarity of sessions. The best results were achieved when the sessions were encoded as binary vectors of the item space

[1] https://github.com/hidasib/GRU4Rec

and when using cosine similarity. In our implementation, the set operations, similarity computations, and the final predictions can be done very efficiently as will be discussed later in Section 4.2.2. Our algorithm has only two parameters, the number of neighbors k and the number of sampled sessions m. For the large e-commerce dataset used in [16], the best parameters were, for example, achieved with $k = 500$ and $m = 1000$. Note that the kNN method used in [16] is based on item-to-item similarities while our kNN method aims to identify similar sessions.

3.1.3 kNN Temporal Extension – TKNN. The KNN method, when using cosine similarity as a distance measure, does not consider the temporal sequence of the events in a session. To be able to leverage the temporal information within the KNN technique, we designed an additional temporal-filtering heuristic for it. The proposed TKNN method uses the same scoring scheme as the KNN method (Equation 1). The only difference is that, given the current session s, we consider item i as being recommendable only if it appears in the neighbor session n directly after a certain item. In our implementation, that certain item is the last item of the current session s. Technically, we therefore use a slightly different implementation of the indicator function of Equation 1: $1_n(i) = 1$ if neighbor session n contains i and (j, i) is a subsequence of n, where j is the last item of the current session and thus the basis to predict the next item.

3.1.4 Simple Association Rules – AR. To assess the strength of simple two-element co-occurrence patterns, we included a method named AR which can be considered as an association rule technique with a maximum rule size of two. Technically, we create a rule $r_{p,q}$ for every two items p and q that appear together in the training sessions. We determine the *weight*, $w_{p,q}$, of each rule simply as the number of times p and q appear together in past sessions. Given the current session s, the AR score of a target item i will be then computed as

$$score_{\text{AR}}(i, s) = w_{i,j} \times 1_{AR}(r_{i,j}) \quad (2)$$

where j is the last item of the current session s for which we want to predict the successor and AR is the set of rules and their weights as determined based on the training data. The indicator function $1_{AR}(r_{i,j}) = 1$ when AR contains $r_{i,j}$ and 0 otherwise.

3.1.5 Simple Sequential Rules – SR. The SR method is a variant of AR, which aims to take the order of the events into account. Similar to the AR method, we create a sequential rule for the co-occurrence of every two items p and q as $r_{p,q}$ in the training data. This time, however, we consider the distance between p and q in the session when computing the weight of the rules. In our implementation, we use the multiplicative inverse as a weight function and set $w_{p,q} = 1/x$, where x is the number of items that appear between p and q in a session. Other heuristics such as a linear or a logarithmic function can also be used. In case that those two items appear together in another session in the training data, the weight of the rule in that session will be added to the current weight. We finally normalize the weight and divide it by the total number of sessions that contributed to the weight. Given the current session s, the SR score of a target item i is then computed as

$$score_{\text{SR}}(i, s) = w_{j,i} \times 1_{SR}(r_{j,i}) \quad (3)$$

Table 1: Dataset characteristics.

	RSC	TMall	#nowplaying	30Music	AotM	8tracks
Sessions	8M	4.6M	95K	170K	83K	500K
Events	32M	46M	1M	2.9M	1.2M	5.8M
Items	38K	620K	115K	450K	140K	600K
Avg. E/S	3.97	9.77	10.37	17.03	14.12	11.40
Avg. I/S	3.17	6.92	9.62	14.20	14.11	11.38

where j is the last item of session s and SR is the set of sequential rules. The indicator function $1_{SR}(r_{j,i}) = 1$ when SR contains $r_{j,i}$ and 0 otherwise.

3.1.6 Hybrid Approaches. We made additional experiments with several hybrids that combine different algorithms. At the end, a weighted combination of the two normalized prediction scores of the algorithms led to the best results in our experiments.

3.2 Datasets and Evaluation Protocol

We performed experiments using datasets from two different domains in which session-based recommendation is relevant, namely e-commerce and next-track music recommendation. The source code and the public datasets can be found online.[2]

3.2.1 E-commerce Datasets. For the e-commerce domain, we chose the *ACM RecSys 2015 Challenge* dataset (RSC) as used in [16]. The RSC dataset is a collection of sequences of click events in shopping sessions. The second e-commerce dataset is a public dataset published for the *TMall* competition. This dataset contains shopping logs of the users on the Tmall.com website.

3.2.2 Music Datasets. We used (a) two datasets that contain *listening logs* of several thousand users and (b) two datasets that comprise thousands of manually created *playlists*.

Listening logs: These used datasets are (almost one-year-long) sub-samples of two public datasets. First, we created a subset of the *#nowplaying* dataset [48], which contains music-related tweets on Twitter. Second, we used the recent *30Music* dataset [41], which contains listening sessions retrieved from Internet radio stations through the Last.fm API.

Playlists: Generally, music playlists are different in nature from listening logs and e-commerce user logs in various ways. Nonetheless, they are designed to be consumed in a listening session and the tracks are often arranged in a specific sequence. The used playlist datasets come from two different music platforms. The *Art-of-the-Mix* dataset (AotM) was published by [31] and contains playlists by music enthusiasts. The *8tracks* dataset was shared with us by the 8tracks platform. A particularity of the 8tracks dataset is that each public playlist can only contain two tracks per artist.

The dataset statistics are shown in Table 1. The total number of sessions is larger for the e-commerce datasets. However, the number of unique items in the music datasets, which corresponds to the number of tracks included in the playlists or the number of played tracks in the listening sessions, is higher than the number of items in e-commerce datasets.

[2] http://ls13-www.cs.tu-dortmund.de/homepage/rectemp2017

The music sessions are on average longer than the e-commerce sessions.[3] The last row of Table 1 shows the average number of unique items in each session ("Avg. I/S"). Comparing this value with the average session length ("Avg. E/S") indicates what we call the item *repetition rate* in each dataset. Including the same track more than once in a playlist is comparably uncommon. Listening to a track more than once during a listening session is, however, common. The difference between the average session length and the average number of items in each session for the e-commerce dataset indicates that re-occurring of the same item in a session is common in the e-commerce domain.

3.2.3 Evaluation Protocol. The general task of the session-based recommendation techniques in our experiment is to predict the next-item view event in a shopping session or to predict the next track that is played in a listening session or is included in a playlist. To evaluate the session-based algorithms, we use the same evaluation scheme as in [16]. We incrementally add events to the sessions in the test set and report the average hit rate (HR), which corresponds to recall in this evaluation setting, and the mean reciprocal rank (MRR), which takes the position of the hit into account. We tested list lengths of 1, 2, 3, 5, 7, 10, 15, and 20. While the experiments in [16] are done without cross-validation, we additionally apply a fivefold sliding-window validation protocol as in [24] to minimize the risk that the obtained results are specific to the single train-test split. We, therefore, created five train-test splits for each dataset. For the listening logs, we used 3 months of training data and the next 5 days as the test data and randomized splits for the playlists as they have no timestamps assigned.

4 RESULTS

4.1 Accuracy Results

Our first experiment used the exact same setup as described in [16], i.e., we use only one training-test split when comparing GRU4REC with our methods. As done in [16], we trained the algorithms using 6 months of data containing 7,966,257 sessions of 31,637,239 clicks on 37,483 items and tested them on the sessions of the next day.

In the subsequent sections, we then report the results of our comparison using the sliding-window validation scheme described above with recommendation list lengths varying from 1 to 20. In all experiments, we tuned the parameters for the different algorithms using grid search and optimized for HR@20 on validation sets (subsets of the training sets). GRU4REC was only optimized with 100 layers as done in [16] due to the computational complexity of the method. To test for statistical significance, we use Wilcoxon signed-rank test with $\alpha = 0.05$.

4.1.1 Results Using the Original Evaluation Setup. Table 2 shows the results ordered by the hit rate (HR@20) when using the original setup. We could reproduce the hit rate and MRR results from [16] (using their optimal parameters) for GRU4REC(1000,BPR) and GRU4REC(1000,TOP1), which use 1000 hidden units and the TOP1 and BPR's pairwise ranking loss functions, respectively. In Table 2, we additionally report the results for recommendation list length ten, which might be more important for different application domains.

[3]Note that each session in the TMall dataset is defined as the sequence of actions of a user during one day which results in relatively larger average session length.

Table 2: Results when using the evaluation scheme of [16].

Method	HR@10	MRR@10	HR@20	MRR@20
SR	**0.568**	**0.290**	**0.672**	**0.297**
TKNN	0.545	0.251	0.670	0.260
AR	0.543	0.273	0.655	0.280
KNN	0.521	0.242	0.641	0.250
GRU4REC(1000,BPR)	0.517	0.235	0.636	0.243
GRU4REC(1000,TOP1)	0.517	0.261	0.623	0.268

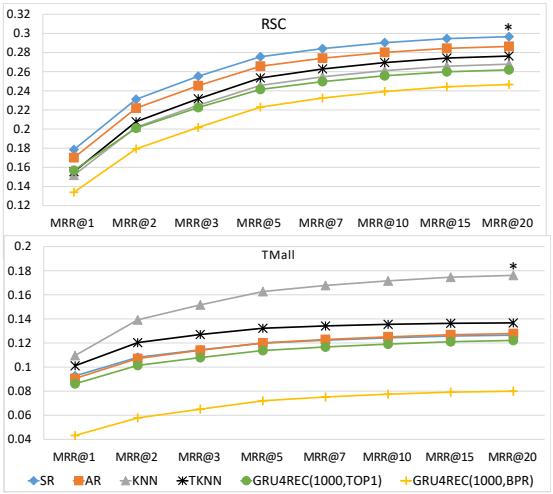


Figure 1: MRR results for the e-commerce datasets (* indicates statistical significance).

The best accuracy results were achieved by the SR method both for the hit rate and MRR and for both list lengths. In terms of the hit rate, every single frequent pattern method used in the experiment was better than the GRU4REC methods. A similar observation can be made also for the MRR, with the exception that the KNN-based methods consistently performed worse than the GRU4REC(1000,TOP1) method on this measure.

4.1.2 E-commerce Datasets. Figure 1 shows the MRR results for the algorithms on the two e-commerce datasets, RSC and TMall. For both datasets, we can observe that most of the frequent pattern methods lead to higher or at least similar MRR values as GRU4REC. There is, however, no clear "winner" across both datasets. The SR method works best for the RSC dataset. On the TMALL dataset, the KNN method outperforms the others, an effect which might be caused by the longer list session lengths for this dataset.[4] In both cases, however, the difference between the winning method and the best-performing GRU4REC configuration is statistically significant. This is indicated by a star symbol in Figure 1.

4.1.3 Listening Logs Datasets. Figure 2 shows the accuracy performance of the algorithms on two selected listening logs datasets.

[4]Remember that the sessions of the TMALL dataset cover the events of one day, as the time stamps in this dataset are given only in the granularity of days.

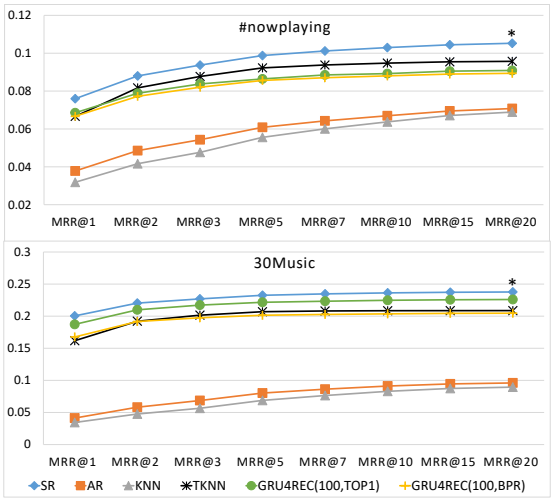


Figure 2: MRR results for the listening log datasets.

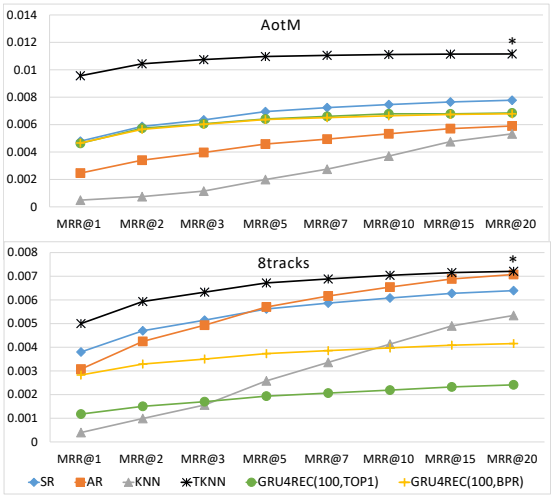


Figure 3: MRR results for the playlist datasets.

Similar to the e-commerce datasets, in all measurements, a frequent pattern approach, namely the SR method, outperforms GRU4REC. Here again, for MRR@20, the recommendations of SR are significantly more accurate than the recommendations of GRU4REC. Note that on the music datasets, we apply GRU4REC(100,TOP1) and GRU4REC(100,BPR), which use 100 hidden units and the TOP1 and BPR's pairwise ranking loss function, respectively.[5]

The TKNN method – the time-aware extension of KNN– works always significantly better than the KNN method on the listening logs datasets. TKNN also outperforms both GRU4REC configurations on the #nowplaying dataset for list lengths larger than 1.

Another observation on the listening logs datasets is that the sequence-based approaches (SR, TKNN and GRU4REC) work significantly better than methods that do not consider the temporal information in data (KNN and AR).

4.1.4 Playlists Datasets. Figure 3 shows the MRR results of the algorithms on the playlists datasets. On both datasets, the temporal extension of kNN, TKNN, leads to the best results across all recommendation list sizes and significantly outperforms both variants of GRU4REC. The performance of all other methods, however, seems to depend on the specifics of the datset. The SR method works well on both datasets. The relative performance of the AR method, however, depends on the dataset and the list length at which the measurement is made.

One interesting observation that we made for the music datasets is that the relative performance of KNN strongly improves in terms of the hit rate[6] when the recommendation list length is increased. This can, for example, be seen in Figure 4, which shows the hit rate results for the #nowplaying dataset. The hit rate of KNN on the #nowplaying dataset that is about 3% for list length one increases

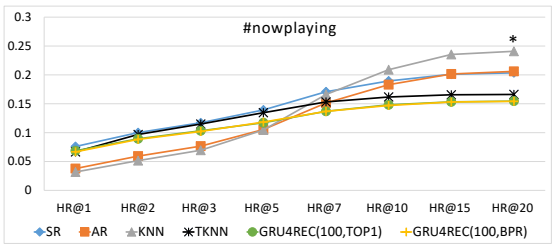


Figure 4: HR results for the #nowplaying dataset.

to 24% for list length 20. At the same time, the hit rate of some of the other methods only slightly increases, e.g., from 6% to 15%. As a result, across all four investigated music datasets, KNN outperforms all other algorithms in terms of HR@20. A similar trend can also be seen for AR, the other non-sequential approach.

4.1.5 Aggregated Ranking of Algorithms. To determine the ranking of different algorithms based on their accuracy results (MRR@20) across all six datasets, we applied the Borda Count (BC) rank aggregation strategy [10]. The results show that SR and TKNN are both ranked first (30 points), followed by AR as the second best algorithm (20 points). The GRU4REC method with TOP1 ranking loss is ranked third (18 points). Finally, KNN and GRU4REC with BPR ranking loss are ranked fourth (15 points) and fifth (13 points), respectively.

4.1.6 Hybrid Approaches. We conducted a variety of additional experiments with different hybridization methods as described in Section 3.1.6 to analyze the effect of combining the algorithms. In general, a weighted combination of the two normalized prediction scores of a neighborhood-based and a sequence-based method led to the best results in our experiments. For instance, the combination of KNN and SR with a weight ratio of 3 to 7, WH(KNN,SR:0.3,0.7), outperformed all other individual algorithms on the 30Music dataset.

[5]Repeating the experiments with 1000 hidden layers for the GRU4REC methods did not lead to any better results on the music datasets.
[6]Generally, the hit rate results for the experiments, which we do not include here for space reasons, are similar to the MRR results.

Table 3: Results of the hybrid methods for 30Music.

Method	HR@5	MRR@5	HR@20	MRR@20
SR	0.285	0.233	0.332	0.238
KNN	0.142	0.069	0.342	0.089
GRU	0.275	0.222	0.315	0.226
WH(KNN,SR:0.3,0.7)	**0.298**	**0.243**	0.386	**0.252**
WH(KNN,GRU:0.6,0.4)	0.261	0.144	**0.396**	0.159

Another example is combining the normalized score of KNN and GRU4REC(100,TOP1), which can outperform other algorithms in terms of HR@20. The differences between the winning hybrid approaches (printed in bold face in Table 3) and the best performing individual methods in each measurement were statistically significant. Similar results were also achieved for the other datasets, which we do not include here for space reasons.

4.2 Additional Analyses

Since prediction accuracy might not be the only possible relevant quality criterion in a domain [19], we made a number of additional analyses as shown in Figure 5.

4.2.1 Popularity Bias and Catalog Coverage. As in [22], we first measured the average popularity of the top-20 recommendations of the algorithms to assess possible recommendation biases. The popularity of an item is computed based on its number of occurrences in the training dataset. Overall, the recommendations of non-sequential approaches (KNN and AR) shows the highest bias towards popular items. The sequence-based approaches (SR and GRU4REC), in contrast, recommend comparably less popular items.

Additionally, we analyzed the catalog coverage of each algorithm by counting the number of different items that appear in the top-20 recommendation lists of all sessions in the test set. Overall, the recommendation lists of GRU4REC and SR include more different items than the other algorithms. The recommendations of neighborhood methods, KNN and TKNN, on the other hand, focus on smaller sets of items and show a higher concentration bias. This can be explained by considering the sampling strategy of KNN which focuses on a smaller subset of the sessions, e.g., those of the last few days.

4.2.2 Computational Complexity and Memory Usage. We measured the training time as well as the needed memory and time to generate recommendations for each algorithm. On a desktop computer with an Intel i7-4790k processor, training GRU4REC on one split of the RSC dataset with almost 4 million sessions and in its best configuration takes more than 12 hours. This time can be reduced to 4 hours when calculations are performed by the GPU (Nvidia GeForce GTX 960).[7] The KNN method needs about 27 seconds to build the needed in-memory maps, see Section 3.1.2. The well-performing SR method needs about 48 seconds to determine the rule weights. A specific advantage of the latter two methods is that they support incremental updates, i.e., new events can be immediately incorporated into the algorithms. Creating one recommendation list with GRU4REC needed, on average, about 12 ms. KNN needs about 26 ms for this task and SR only 3 ms.

[7] Training the model for 6 month of data using the GPU lasts about 8 hours.

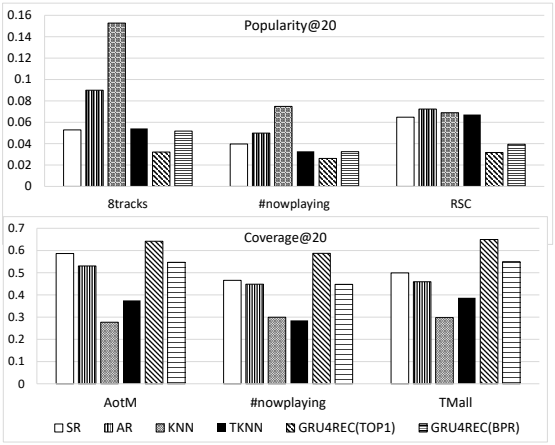


Figure 5: Popularity biases and catalog coverages of the algorithms on three sample datasets.

The raw data used for training the algorithms in this specific experiment (one split of the RSC dataset) occupies about 540 MB of main memory. The data structures used for training SR and KNN occupy about 50 MB and 3.2 GB, respectively. The model created by GRU4REC needs about 510 MB. Note that memory demand of GRU4REC depends on the algorithm parameters and significantly increases with the number of items. For the music and Tmall datasets, the memory demand of GRU4REC exceeded the capacity of our graphics card. Running GRU4REC using the CPU is multiple times slower than when a graphics card is used.

5 CONCLUSION AND FUTURE WORKS

Our work indicates that comparably simple frequent-pattern-based approaches can represent a comparably strong baseline when evaluating session-based recommendation problems. At the end, we could find at least one pattern-based approach that was significantly better than a recent RNN-based method. In particular the SR method was surprisingly effective, despite the fact that both learning and applying the rules is very fast.

Our results also indicates that the "winning" strategy seems to strongly depend on the characteristics of the data sets like average session lengths or repetition rates. Further research is still required to understand this relationship. In our future work, we will investigate the performance of additional session-based algorithms. These algorithms include both ones that are based on Markov models, e.g., Rendle et al.'s factorized Markov chains [34], as well as recently proposed improvements to GRU4REC, e.g., by Tan et al. [38]. We expect that continuously improved RNN-based methods will be able to outperform the frequent pattern based baselines used in the evaluation reported in this paper. These methods can, however, be computationally quite expensive. From a practical perspective, one has therefore to assess depending on the application domain if the obtained gains in accuracy justify the usage of these complex models, which cannot be easily updated online and whose predictions can be difficult to explain.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining Association Rules between Sets of Items in Large Databases. In *SIGMOD '93*. 207–216.

[2] Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining Sequential Patterns. In *ICDE '95*. 3–14.

[3] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-task Learning for Deep Text Recommendations. In *RecSys '16*. 107–114.

[4] Geoffray Bonnin and Dietmar Jannach. 2014. Automated Generation of Music Playlists: Survey and Experiments. *ACM Computing Surveys* 47, 2 (2014), 26:1–26:35.

[5] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *CoRR* abs/1412.3555 (2014).

[6] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *RecSys '16*. 191–198.

[7] Sander Dieleman. 2016. Deep Learning for Audio-Based Music Recommendation. In *DLRS '16 Workshop*. 1–1.

[8] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems. In *WWW '15*. 278–288.

[9] Jeffrey L. Elman. 1990. Finding Structure in Time. *Cognitive Science* 14, 2 (1990), 179 – 211.

[10] Peter Emerson. 2013. The Original Borda Count and Partial Voting. *Social Choice and Welfare* 40, 2 (2013), 353–358.

[11] Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *CoRR* abs/1308.0850 (2013). http://arxiv.org/abs/1308.0850

[12] Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural Turing Machines. *CoRR* abs/1410.5401 (2014).

[13] Yupeng Gu, Bo Zhao, David Hardtke, and Yizhou Sun. 2016. Learning Global Term Weights for Content-based Recommender Systems. In *WWW '16*. 391–400.

[14] Jiawei Han and Micheline Kamber. 2006. *Data Mining: Concepts and Techniques (Second Edition)*. Morgan Kaufmann.

[15] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-Aware Music Recommendation Based on Latent Topic Sequential Patterns. In *RecSys '12*. 131–138.

[16] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *CoRR* abs/1511.06939 (2015).

[17] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations. In *RecSys '16*. 241–248.

[18] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (Nov. 1997), 1735–1780.

[19] Dietmar Jannach and Gedas Adomavicius. 2016. Recommendations with a Purpose. In *RecSys '16*. 7–10.

[20] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. 2015. Adaptation and Evaluation of Recommendations for Short-term Shopping Goals. In *RecSys '15*. 211–218.

[21] Dietmar Jannach, Lukas Lerche, and Iman Kamehkhosh. 2015. Beyond "Hitting the Hits": Generating Coherent Music Playlist Continuations with the Right Tracks. In *RecSys '15*. 187–194.

[22] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. 2015. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction* (2015), 1–65.

[23] Dietmar Jannach and Malte Ludewig. 2017. Determining Characteristics of Successful Recommendations from Log Data – A Case Study. In *SAC '17*.

[24] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation. In *RecSys 2017*. (forthcoming).

[25] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. 2016. Convolutional Matrix Factorization for Document Context-Aware Recommendation. In *RecSys '16*. 233–240.

[26] Lukas Lerche, Dietmar Jannach, and Malte Ludewig. 2016. On the Value of Reminders within E-Commerce Recommendations. In *UMAP '16*. 27–25.

[27] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep Collaborative Filtering via Marginalized Denoising Auto-encoder. In *CIKM '15*. 811–820.

[28] Zachary Chase Lipton. 2015. A Critical Review of Recurrent Neural Networks for Sequence Learning. *CoRR* abs/1506.00019 (2015).

[29] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. In *SIGIR '15*. 43–52.

[30] Brian McFee and Gert R. G. Lanckriet. 2011. The Natural Language of Playlists. In *ISMIR '11*. 537–542.

[31] Brian McFee and Gert R. G. Lanckriet. 2012. Hypergraph Models of Playlist Dialects. In *ISMIR '12*. 343–348.

[32] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. 2002. Using Sequential and Non-Sequential Patterns in Predictive Web Usage Mining Tasks. In *ICDM '02*. 669–672.

[33] Ozlem Ozgobek, Jon A. Gulla, and Riza C. Erdur. 2014. A Survey on Challenges and Methods in News Recommendation. In *WEBIST '14*. 278–285.

[34] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *WWW '10*. 811–820.

[35] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *The Journal of Machine Learning Research* 6 (Dec. 2005), 1265–1295.

[36] Yang Song, Ali Mamdouh Elkahky, and Xiaodong He. 2016. Multi-Rate Deep Learning for Temporal Recommendation. In *SIGIR '16*. 909–912.

[37] Florian Strub, Romaric Gaudel, and Jérémie Mary. 2016. Hybrid Recommender System based on Autoencoders. In *DLRS '16 Workshop*. 11–16.

[38] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved Recurrent Neural Networks for Session-based Recommendations. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS '16)*. ACM, 17–22.

[39] Maryam Tavakol and Ulf Brefeld. 2014. Factored MDPs for Detecting Topics of User Sessions. In *RecSys '14*. 33–40.

[40] Roberto Turrin, Andrea Condorelli, Paolo Cremonesi, Roberto Pagano, and Massimo Quadrana. 2015. Large Scale Music Recommendation. In *LSRS 2015 Workshop at ACM RecSys*.

[41] Roberto Turrin, Massimo Quadrana, Andrea Condorelli, Roberto Pagano, and Paolo Cremonesi. 2015. 30Music Listening and Playlists Dataset. In *Poster Proceedings RecSys '15*.

[42] Bartłomiej Twardowski. 2016. Modelling Contextual Information in Session-Aware Recommender Systems with Neural Networks. In *RecSys '16*. 273–276.

[43] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation. In *RecSys '16*. 225–232.

[44] Koen Verstrepen and Bart Goethals. 2014. Unifying Nearest Neighbors Collaborative Filtering. In *RecSys '14*. 177–184.

[45] Jeroen B. P. Vuurens, Martha Larson, and Arjen P. de Vries. 2016. Exploring Deep Space: Learning Personalized Ranking in a Semantic Space. In *DLRS '16 Workshop*. 23–28.

[46] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *KDD '15*. 1235–1244.

[47] Ghim-Eng Yap, Xiao-Li Li, and Philip S. Yu. 2012. Effective Next-items Recommendation via Personalized Sequential Pattern Mining. In *DASFAA'12*. Berlin, Heidelberg, 48–64.

[48] Eva Zangerle, Martin Pichl, Wolfgang Gassler, and Günther Specht. 2014. #Nowplaying Music Dataset: Extracting Listening Behavior from Twitter. In *WISMM '14 Workshop at MM '14*. 21–26.

[49] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. In *AAAI '14*. 1369–1375.

User Modeling and User-Adapted Interaction

Evaluation of Session-based Recommendation Algorithms

Malte Ludewig · Dietmar Jannach

October 2018

Abstract Recommender systems help users find relevant items of interest, for example on e-commerce or media streaming sites. Most academic research is concerned with approaches that personalize the recommendations according to long-term user profiles. In many real-world applications, however, such long-term profiles often do not exist and recommendations therefore have to be made solely based on the observed behavior of a user during an ongoing session. Given the high practical relevance of the problem, an increased interest in this problem can be observed in recent years, leading to a number of proposals for *session-based recommendation algorithms* that typically aim to predict the user's immediate next actions.

In this work, we present the results of an in-depth performance comparison of a number of such algorithms, using a variety of datasets and evaluation measures. Our comparison includes the most recent approaches based on recurrent neural networks like GRU4REC, factorized Markov model approaches such as FISM or FOSSIL, as well as simpler methods based, e.g., on nearest neighbor schemes. Our experiments reveal that algorithms of this latter class, despite their sometimes almost trivial nature, often perform equally well or significantly better than today's more complex approaches based on deep neural networks. Our results therefore suggest that there is substantial room for improvement regarding the development of more sophisticated session-based recommendation algorithms.[1]

Malte Ludewig
TU Dortmund, Germany
E-mail: malte.ludewig@tu-dortmund.de

Dietmar Jannach
AAU Klagenfurt, Austria
E-mail: dietmar.jannach@aau.at

[1] A preliminary comparison of sequential recommendation algorithms was presented in our own previous work in (Jannach and Ludewig, 2017; Kamehkhosh et al., 2017) and a preprint version of this work is available at `https://arxiv.org/abs/1803.09587`. This paper or a similar version is not currently under review by a journal or conference. This paper is

Keywords Session-based Recommendation; Sequential Recommendation; Deep Learning; Factorized Markov Models, Nearest Neighbors

1 Introduction

Many of today's online services use recommender systems to point their users or site visitors to additional items that might be of interest to them. In academic research, the majority of works is focusing on techniques that rely on long-term preference models to determine the items to be presented to the user. However, in many application domains of recommender systems, such long-term user models are often not available for a larger fraction of the users, e.g., because they are first-time visitors or because they are not logged in. Consequently, suitable recommendations have to be determined based on other types of information, usually the user's most recent interactions with the site or application. Recommendation techniques that rely solely on the user's actions in an ongoing session and which adapt their recommendations to the user's actions are called *session-based* recommendation approaches (Quadrana et al., 2018).

Amazon's "Customers who bought ... also bought" recommendations can be considered an extreme case of such a session-based approach. In this case, the recommendations are seemingly only dependent on the item that is currently viewed by the user (and the purchasing patterns of the community). A number of other techniques were proposed in the research literature, which do not limit themselves to the very last action, but consider some or all user actions since the session started. Some of these techniques only consider which events happened; others, in contrast, in addition take the sequence of events into account in their algorithms. Besides the e-commerce domain, a number of other application fields were in the focus in the literature, among them in particular music, web page navigation, or travel and tourism.

In academia, sequential recommendation problems are typically operationalized as the task of predicting the next user action. Experimental evaluations are usually based on larger, time-ordered logs of user actions, e.g., on the users' item viewing and purchase activities on an e-commerce shop or on their listening history on a music streaming site. From an algorithmic perspective, early approaches to predict the next user actions were based, for example, on sequential pattern mining techniques. Later on, different types of more sophisticated methods based on Markov models were proposed and successfully applied to the problem. Finally, in the most recent years, the use of deep learning approaches based on artificial neural networks was explored as another solution. Recurrent Neural Networks (RNN), which are capable of learning models from sequentially ordered data, are a "natural choice" for this problem, and significant advances regarding the prediction accuracy of such algorithms were reported in the recent literature (Hidasi et al., 2016a; Tan

void of plagiarism or self-plagiarism as defined by the Committee on Publication Ethics and Springer Guidelines.

et al., 2016; Hidasi et al., 2016b; Hidasi and Karatzoglou, 2017; Devooght and Bersini, 2017).

Despite the growing number of papers on the topic in recent years, no true "standard" benchmark data sets or evaluation protocols exist in the community. Therefore, it remains difficult to compare the various algorithmic proposals, in particular as often different baseline algorithms are used in the papers. And, for some of them it is also unclear if they are particularly strong. In our previous work (Jannach and Ludewig, 2017; Kamehkhosh et al., 2017), we could, for example, demonstrate that a comparably simple k-nearest-neighbor method leads to similar or even better accuracy results than a modern deep learning approach.

To establish a common base for future research, we performed an in-depth performance comparison across multiple domains and datasets, which involved a number of comparably simple as well as more sophisticated algorithms from the recent literature. Our results show that computationally and conceptually simple methods often lead to predictions that are similarly accurate or even better than those of today's most recent techniques based on deep learning models. As a consequence, we argue that researchers should take these simpler methods as alternative baselines into account when developing novel session-based recommendation algorithms. Furthermore, our results suggest that there is still substantial room for improvement regarding the development of more sophisticated session-based recommendation algorithms.

This paper extends our previous works presented in (Jannach and Ludewig, 2017; Kamehkhosh et al., 2017) in a number of ways. First, we made experiments for a larger number of datasets from different domains, using a richer set of performance measures. Second, we included recent *sequential* recommendation algorithms like FISM and FOSSIL (Kabbur et al., 2013; He and McAuley, 2016) in the evaluation as well as the latest version of GRU4REC (Hidasi et al., 2016a). Third, we designed a number of additional sequence-aware similarity measures for the previously proposed session-based nearest neighbor method, which in most cases lead to significant performance gains. Finally, we also propose a new method called *Session-based Matrix Factorization (SFM)*, which yields good results in some of the tested application domains.

The paper is organized as follows. Next, in Section 2, we discuss previous works and typical application areas of session-based recommendation approaches. In Section 3, we provide technical details about the algorithms that were compared in our work. Section 4 describes our evaluation setup and Section 5 the outcomes of our experiments. To foster reproducible research on the topic, we share the code of the used evaluation framework and the compared algorithms online.[2]

[2] https://www.dropbox.com/sh/dbzmtq4zhzbj5o9/AACldzQWbw-igKjcPTBI6ZPAa?dl=0

2 Review of Session-Based Recommendation Approaches

Most of the approaches for session-based recommendation proposed in the literature implement some form of *sequence learning*, see also (Quadrana et al., 2018) for a recent survey on the more general class of sequence-aware recommenders. Early approaches were based on the identification of *frequent sequential patterns*, which can be used at recommendation time to predict a user's next action. These early approaches were applied, for example, in the context of predicting the online navigation behavior of users (Mobasher et al., 2002). Later on, such pattern mining techniques were also used for next-item recommendation problems in e-commerce or the music domain (Yap et al., 2012; Hariri et al., 2012; Bonnin and Jannach, 2014).

While frequent pattern techniques are easy to implement and lead to interpretable models, the mining process can be computationally demanding. At the same time, finding good algorithm parameters, in particular a suitable minimum support threshold, can be challenging. Finally, in some application domains it seems that using frequent item sequences does not lead to better recommendations than when using simpler item co-occurrence patterns (Bonnin and Jannach, 2014). In the context of this work, we investigate both sequential and co-occurrence patterns in their simplest forms as baselines.

In many newer works, more sophisticated sequence learning approaches were proposed that implement some form of *sequence modeling*. Such sequence modeling approaches are usually based on Markov Chain (MC) models (He et al., 2009; Mcfee and Lanckriet, 2011; Garcin et al., 2013; Hosseinzadeh Aghdam et al., 2015), reinforcement learning (RL) and Markov Decision Processes (MDP) (Shani et al., 2005; Moling et al., 2012; Tavakol and Brefeld, 2014), or Recurrent Neural Networks (RNN) (Zhang et al., 2014; Sordoni et al., 2015; Hidasi et al., 2016a,b; Liu et al., 2016; Song et al., 2016; Twardowski, 2016; Yu et al., 2016; Du et al., 2016; Soh et al., 2017). Again, the typical application scenarios of these methods include the e-commerce and the music recommendation domain.

An early approach based on an MDP model was proposed by Shani et al. (2005). It demonstrated the value of using sequential data in an e-commerce scenario, but also showed that models based on Markov Chains often cannot be directly applied due to data sparsity. Therefore, Shani et al. (2005) proposed different heuristics to overcome the problem. An additional challenge when using this type of models is to decide how many preceding interactions should be considered when predicting the next one. Some authors therefore use a mixture of Variable-order Markov Models (VMMs) or *context-trees* to consider sequences of different lengths (He et al., 2009; Garcin et al., 2013). Other works, for example by Hosseinzadeh Aghdam et al. (2015), rely on Hidden Markov Models (HMMs) to overcome certain limitations of plain Markov Chain models. In (Shani et al., 2005; Moling et al., 2012), reinforcement learning was implemented based on MDPs, which made it possible to also consider the reward for the shop in the recommendation process. To deal with the problem of the explosion of the state space in such scenarios, Tavakol and

Brefeld (2014) proposed to model the state space based on the sequence of item attributes in order to predict the characteristics of the next item that the user will consider. In the context of the comparative analysis presented in this paper, we limit ourselves to a simple MC-based method as a baseline, in particular because some techniques like the one discussed by Tavakol and Brefeld (2014) require the existence of knowledge about certain item attributes.

The most recent works on sequence modeling are based on RNNs. Zhang et al. (2014), for example, used them for the prediction of user clicks in an advertisement scenario. Hidasi et al. (2016a) were among the first to explore Gated Recurrent Units (GRUs) as a special form of RNNs for the prediction of the next user action in a session. Their method called GRU4REC was later on extended in different ways in (Hidasi et al., 2016b; Hidasi and Karatzoglou, 2017) and (Quadrana et al., 2017). While Hidasi et al. (2016a) reported substantial performance improvements over an *item-based* k-nearest-neighbor (kNN) method when using their first version of GRU4REC, our previous work (Jannach and Ludewig, 2017) showed that a *session-based* nearest neighbor method also leads to competitive accuracy results for the same problem setting. Since GRU4REC was substantially improved since its initial version, we include the latest version of the method proposed by Hidasi and Karatzoglou (2017) in the performance comparison reported in this paper. Furthermore, given our observations regarding the often competitive performance of conceptually simpler methods we designed a number of variations of the basic session-based nearest neighborhood method from (Jannach and Ludewig, 2017), which we also considered in the experiments.

Another family of sequence modeling approaches relies on *distributed item representations*, e.g., in the form of latent Markov embeddings (Chen et al., 2012, 2013; Wu et al., 2013; Feng et al., 2015) or distributional embeddings (Djuric et al., 2014; Baeza-Yates et al., 2015; Grbovic et al., 2015; Tagami et al., 2015; Vasile et al., 2016; Reddy et al., 2016; Zheleva et al., 2010). Embeddings are dense, lower-dimensional representations that are derived from sequentially ordered data and encode transition probabilities based on the observations in the original data. They were applied, for example, in the domains of next-track music recommendation (Zheleva et al., 2010; Chen et al., 2012), recommendation of learning courses (Reddy et al., 2016), or next point-of-interest (POI) recommendation (Feng et al., 2015). However, a general challenge when using item embeddings is that they can be computationally demanding and sometimes require substantial amounts of training data to be effective. In the context of our work, we experimented with item embeddings as an alternative representation of the user sessions. However, the usage of embeddings did not lead to an improvement in terms of the prediction accuracy for our problem settings, which is why we do not report the detailed outcomes of these experiments in this paper.

To overcome the limitations of pure sequence learning methods, a number of *hybrid methods* were proposed that, for instance, combine the advantages of matrix factorization techniques with sequence modeling approaches in the form of Factorized Markov Chains (Rendle et al., 2010; Lian et al., 2013; Cheng

et al., 2013; He et al., 2016; He and McAuley, 2016). Rendle et al. (2010) proposed the Factorized Personalized Markov Chain (FPMC) approach as an early method for next-item recommendations in e-commerce settings, where user interactions are represented as a three-dimensional tensor (user, current item, next-item). Later on, variations of FPMC were proposed and successfully applied for a variety of application problems, e.g., by Kabbur et al. (2013) and He and McAuley (2016). Other hybrid techniques that, for example, use some form of clustering or Latent Dirichlet Allocation in combination with a sequential recommendation method were proposed, e.g., in (Hariri et al., 2012; Natarajan et al., 2013; Song et al., 2015), for the problems of next-track or next-app recommendation. In our experimental evaluation, we include both the FPMC method by Rendle et al. (2010) as well as the recent variations and improvements described by Kabbur et al. (2013) (FISM) and He and McAuley (2016) (FOSSIL).

Besides pure *session-based* techniques, which solely consider a user's action of the ongoing session, there are also approaches that consider previous interactions of the same user in the recommendation process. Such techniques are called *session-aware* according to the terminology of Quadrana et al. (2018). Examples of such works include (Baeza-Yates et al., 2015; Billsus et al., 2000; Hariri et al., 2012; Jannach et al., 2015a, 2017a; Quadrana et al., 2017), and session-aware approaches were applied for various application domains like e-commerce, music, news, or next-app recommendation. Considering longer-term user preferences in these papers shows to be helpful to improve the recommendations in the current, ongoing session. In some cases, like in (Jannach et al., 2015a), it however turns out that the short-term user intents are much more important than the longer-term models. In the research presented herein, we therefore exclusively focus on session-based recommendation scenarios. We however consider the combination of long-term and short-term models as an important area for future research.

3 Details of the Investigated Methods

Based on these discussion, we include the following four types of techniques in our comparison of session-based recommendation algorithms: simple heuristics as baseline methods, nearest-neighbor techniques, recurrent neural networks, and factorization-based methods. The main input to all methods is a training set of past user sessions, where each session consists of a set of sequentially ordered actions of a given type, e.g., an item view event in an online shop or a consumption event on a media streaming site. The models learned by the algorithms can then be used to predict the next event in a given user session in the test set. In our evaluations, we follow a pragmatic approach to determine user sessions—in case these are not provided in the datasets—and use user inactivity times to determine session borders. The details for each dataset are described later in this paper.

Regarding the choice of the algorithms, we focus on collaborative filtering methods based on implicit feedback signals, e.g., item view or music listening events. Depending on the specific application, content-based and hybrid algorithms can be designed that use additional meta-data or content features. Since these features are domain specific and such features are only available for very few of our datasets, we limit ourselves to methods that do not rely on such types of data in this paper.

3.1 Baseline Methods

We include the following baseline techniques in our comparison: a method that we call Simple Association Rules (AR), first-order Markov Chains (MC), and a method that we named Sequential Rules (SR). All baselines implement very simple prediction schemes, have a low computational complexity both for training and recommending, and only consider the very last item of a current user session to make the predictions. Furthermore, we include a prediction method based on Bayesian Personalized Ranking (BPR-MF) proposed by Rendle et al. (2009) as an alternative baseline.

3.1.1 Simple Association Rules (AR)

Simple Association Rules (AR) are a simplified version of the association rule mining technique (Agrawal et al., 1993) with a maximum rule size of two. The method is designed to capture the frequency of two co-occurring events, e.g., "Customers who bought ... also bought". Algorithmically, the rules and their corresponding importance are "learned" by counting how often the items i and j occurred together in a session of any user.

Let a session s be a chronologically ordered tuple of item click events $s = (s_1, s_2, s_3, \ldots, s_m)$ and S_p the set of all past sessions. Given a user's current session s with $s_{|s|}$ being the last item interaction in s, we can define the score for a recommendable item i as follows, where the indicator function $1_{\text{EQ}}(a, b)$ is 1 in case a and b refer to the same item and 0 otherwise.

$$score_{\text{AR}}(i, s) = \frac{1}{\sum_{p \in S_p} \sum_{x=1}^{|p|} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot (|p| - 1)} \sum_{p \in S_p} \sum_{x=1}^{|p|} \sum_{y=1}^{|p|} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot 1_{\text{EQ}}(i, p_y) \tag{1}$$

In Equation 1, the sums at the right-hand side represent the counting scheme. The term at the left-hand side normalizes the score by the number of total rule occurrences originating from the current item $s_{|s|}$. A list of recommendations returned by the AR method then contains the items with the highest scores in descending order. No minimum support or confidence thresholds are applied. In our implementation, as shared online, we create the rules in one iteration over the training data and store them (sorted by weight) in nested maps to support fast lookups in the recommendation phase. With this data structure, top-n recommendations can be created almost instantaneously.

3.1.2 Markov Chains (MC)

The MC baseline can be seen as a variant of AR with a focus on sequences in the data. Here, the rules are extracted from a first-order Markov Chain, see (Norris, 1997), which describes the transition probability between two *subsequent* events in a session. In our baseline approach, we simply count how often users viewed item q immediately after viewing item p. Technically, the score for an item i given the current session s with the last event $s_{|s|}$ can be defined as a simplified version of Equation 1:

$$score_{\text{MC}}(i,s) = \frac{1}{\sum_{p \in S_p} \sum_{x=1}^{|p|-1} 1_{\text{EQ}}(s_{|s|}, p_x)} \sum_{p \in S_p} \sum_{x=1}^{|p|-1} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot 1_{\text{EQ}}(i, p_{x+1}) \quad (2)$$

where the function $1_{\text{EQ}}(a,b)$ again indicates whether a and b refer to the same item or not. Here, with the right-hand side of the formula, we count how often item i appears immediately after $s_{|s|}$. The normalization term transforms the absolute count into a relative transition probability. In line with AR, in our implementation the rules and weights are recorded in nested maps in one single iteration over the training data to ensure short training times and to support the fast generation of the recommendations.

3.1.3 Sequential Rules (SR)

Finally, the SR method as proposed in (Kamehkhosh et al., 2017) is a variation of MC or AR respectively. It also takes the order of actions into account, but in a less restrictive manner. In contrast to the MC method, we create a rule when an item q appeared after an item p in a session even when other events happened between p and q.

When assigning weights to the rules, we consider the number of elements appearing between p and q in the session. Specifically, we use the weight function $w_{\text{SR}}(x) = 1/(x)$, where x corresponds to the number of steps between the two items.[3] Given the current session s, the SR method calculates the score for the target item i as follows:

$$score_{\text{SR}}(i,s) = \frac{1}{\sum_{p \in S_p} \sum_{x=2}^{|p|} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot x} \sum_{p \in S_p} \sum_{x=2}^{|p|} \sum_{y=1}^{x-1} 1_{\text{EQ}}(s_{|s|}, p_y) \cdot 1_{\text{EQ}}(i, p_x) \cdot w_{\text{SR}}(x-y) \quad (3)$$

In contrast to Equation 1 for AR, the third inner sum only considers indices of previous item view events for each session p. In addition, the weighting function $w_{\text{SR}}(x)$ is added. Again, we normalize the absolute score by the total number of rule occurrences for the current item $s_{|s|}$. As for AR and MC, the algorithm was implemented using nested sorted maps, which can be created in a single iteration over the training data.

[3] Other weighting functions, e.g., with a logarithmic decay, are possible as well. Using the linear function however led to the best results, on average, in our experiments.

*3.1.4 Bayesian Personalized Ranking (*BPR-MF*)*

To make our results comparable with previous research, we finally include a prediction method based on BPR-MF as a baseline in our experiments.[4] BPR-MF proposed by Rendle et al. (2009) is a learning-to-rank method designed for implicit-feedback recommendation scenarios. The method is usually applied for matrix-completion problem formulations based on longer-term user-item interactions. In BPR-MF the matrix is factorized into two smaller matrices of latent user and item features (W and H), optimizing the following criterion:

$$BPR_{OPT} = \sum_{(u,i,j) \in D_S} ln\, \sigma(r_{u,i} - r_{u,j}) - \lambda_\Theta ||\Theta||^2 \tag{4}$$

In the above formula, a ranking $r_{u,i}$ for user u and item i is approximated with the dot product of the corresponding rows in the matrices W and H ($r_{u,i} = \langle W_u, H_i \rangle$). The model parameters $\Theta = (W, H)$ are learned using stochastic gradient descent in multiple iterations over the dataset D_S, which consists of triplets of the form (u, i, j), where (u, i) is a positive feedback pair and (u, j) is a sampled negative example. The optimization criterion in Equation 4 aims to rank the positive sample (u, i) higher than a non-observed sample (u, j).

To apply the method for the session-based recommendation scenario—where there are no long-term user profiles—we attribute each session in the training set to a different user, i.e., each session corresponds to a user in the user-item interaction matrix. At prediction time, we use the average of the latent item vectors of the current session so far as the user vector.

Generally, BPR and other methods designed for the matrix-completion problems in their original form, i.e., without considering the short-term session context, do not lead to competitive results in session-based recommendation scenarios, as reported, e.g., in (Jannach et al., 2015a). Therefore, we do not consider such algorithms, e.g., traditional matrix factorization techniques, as baselines in our experiments.

3.2 Nearest Neighbors

Despite their simplicity, nearest-neighbor-based approaches often perform surprisingly well as discussed, e.g., by Verstrepen and Goethals (2014) and in our previous work (Jannach and Ludewig, 2017; Kamehkhosh et al., 2017). We, therefore, include different nearest neighbor schemes in our comparison. First, we consider a more traditional item-based variant, which was also employed as a baseline method by Hidasi et al. (2016a). Furthermore, we evaluate three variations of a more recent session-based nearest neighbor technique in our experiments.

[4] The method was proposed by Hidasi et al. in the context of the GRU4REC method.

*3.2.1 Item-based kNN (*IKNN*)*

The IKNN method as used in (Hidasi et al., 2016a) only considers the last element in a given session and then returns those items as recommendations that are most similar to it in terms of their co-occurrence in other sessions. Technically, each item is encoded as a binary vector, where each element corresponds to a session and is set to "1" in case the item appeared in the session. The similarity of two items can then be determined, e.g., using the cosine similarity measure, and the number of neighbours k is implicitly defined by the desired recommendation list length.

Conceptually, the method implements a certain form of a "Customers who bought ... also bought" scheme like the AR baseline. The use of the cosine similarity metric however makes it less susceptible to popularity biases. Although item-to-item approaches are comparably simple, they are commonly used in practice and sometimes considered a strong baselines (Linden et al., 2003; Davidson et al., 2010). In terms of the technical implementation, all similarity values can be pre-computed and sorted in the training process to ensure fast responses at recommendation time.[5]

*3.2.2 Session-based kNN (*SKNN*)*

Instead of considering only the last event in the current session, the SKNN method compares the entire current session with the past sessions in the training data to determine the items to be recommended, see also (Hariri et al., 2012; Bonnin and Jannach, 2014; Lerche et al., 2016). Technically, given a session s, we first determine the k most similar past sessions (neighbors) N_s by applying a suitable session similarity measure, e.g., the Jaccard index or cosine similarity on binary vectors over the item space (Bonnin and Jannach, 2014). In our experiments, the binary cosine similarity measure led to the best results. As in (Jannach and Ludewig, 2017), using $k = 500$ as the number of neighbors to consider led to good performance results for many datasets. Next, given the current session s, its neighbors N_s, and the chosen similarity function $sim(s_1, s_2)$ for two sessions s_1 and s_2, the recommendation score for each item i can as defined by Bonnin and Jannach (2014):

$$score_{\text{SKNN}}(i, s) = \Sigma_{n \in N_s} sim(s, n) \cdot 1_n(i) \tag{5}$$

Here, the indicator function $1_n(i)$ returns 1 if session n contains item i and 0 otherwise.

Scalability Considerations. Given a current session s, we cannot scan a potentially large set of past sessions for possible neighbors in an online recommendation scenario. Therefore, in our implementation of the algorithm, as described in (Jannach and Ludewig, 2017) in more detail, we rely on pre-computed in-memory index data structures and on neighborhood sampling to enable fast

[5] We use the implementation published at `https://github.com/hidasib/GRU4Rec`.

recommendation responses. The index is used to quickly locate past sessions that contain a certain item, i.e., the index allows us to retrieve *possible* neighbor sessions that contain at least one element of the current session through fast lookup operations. On the other hand, sampling only a smaller fraction of all past sessions in our experiments as potential neighbors has shown to lead to comparably small accuracy compromises. In fact, in some domains like e-commerce, only looking for neighbors in the most recent sessions—thereby capturing recent trends in the community—proved to be very effective (Jannach et al., 2017b) and led to even better results than when all past sessions were taken into account.

Our nearest neighbor implementations, therefore, have an additional parameter m, which determines the size of the sample from which the neighbors of a target session are taken. In the experiments reported in (Jannach and Ludewig, 2017), it was, for example, sufficient to consider only the 1,000 most recent sessions from several million existing ones.

Sequence-Aware Extensions: V-SKNN*,* S-SKNN*, and* SF-SKNN The described SKNN method does not consider the order of the elements in a session when using the Jaccard index or cosine similarity as a distance measure. Since the order of the elements might, however, be relevant in some domains and since the user preferences might change within a single session depending on the already seen items, we propose three variations of the SKNN method.[6]

- Vector Multiplication Session-Based kNN (V-SKNN): The idea of this variant is to put more emphasis on the more recent events of a session when computing the similarities. Instead of encoding a session as a *binary* vector as described above, we use real-valued vectors to encode the current session. Only the very last element of the session obtains a value of "1"; the weights of the other elements are determined using a linear decay function that depends on the position of the element within the session, where elements appearing earlier in the session obtain a lower weight. As a result, when using the *dot product* as a similarity function between the current weight-encoded session and a binary-encoded past session, more emphasis is given to elements that appear later in the sessions.
- Sequential Session-based kNN (S-SKNN): This variant also puts more weight on elements that appear later in the session. This time, however, we achieve the effect with the following scoring function:

$$score_{\text{S-SKNN}}(i, s) = \Sigma_{n \in N_s} sim(s, n) \cdot w_n(s) \cdot 1_n(i) \quad (6)$$

  Here, the indicator function $1_n(i)$ is complemented with a weighting function $w_n(i, s)$, which takes the order of the events in the current session s into account. The weight $w_n(i, s)$ increases when the more recent items of

[6] We made additional experiments using other ways of encoding sequential information, e.g., by using embeddings of sessions and items with the popular *Word2Vec* and *Doc2Vec* approaches. However, none of these variations led to better accuracy results than the SKNN method in our experiments. We therefore omit these results from our later discussions.

the current session s also appeared in a neighboring session n. If an item s_x is the most recent item of the current session s that also appears in the neighbor session n, then the weight will be defined as $w_n(s) = x/|s|$, where the index x indicates the position of s_x within the session.[7] If, for example, the second-to-last item of the current session with a length of 5 is the most recent item also included in the neighbor session n, the weight would be $w_n(i, s) = 4/5$. Items from this neighbor can, therefore, potentially obtain a higher score than, e.g., items from neighbor sessions that only include the third from last item of the current session, which are assigned a weight of 3/5.

– Sequential Filter Session-based kNN (SF-SKNN): This method also uses a modified scoring function, but in a more restrictive way. The basic idea is that given the last event (and related item $s_{|s|}$) of the current session s, we only consider items for recommendation that appeared directly after $s_{|s|}$ in the training data at least once.

$$score_{\text{SF-SKNN}}(i, s) = \Sigma_{n \in N_s} sim(s, n) \cdot 1_n(s_{|s|}, i) \tag{7}$$

While the general scoring function is identical to the one of SKNN (Equation 5), we use a different implementation of the indicator function $1_n(s_{|s|}, i)$. Here, 1 is only returned if there exists any past session which contains the sequence $(s_{|s|}, i)$, given $s_{|s|}$ is the item currently viewed in the user's current session s. Though the sequence $(s_{|s|}, i)$ can be part of any past session, the item i obviously still has to be a part of the neighbor session n for the indicator function to return 1.

3.3 Neural Networks – GRU4REC

Approaches based on Recurrent Neural Networks (RNNs), as discussed in Section 2, represent the most recently explored family of techniques for session-based recommendation problems. Among these methods, GRU4REC is one of the latest deep learning approaches that was specifically designed for session-based recommendation scenarios (Hidasi et al., 2016a; Hidasi and Karatzoglou, 2017).

GRU4REC models user sessions with the help of an RNN with Gated Recurrent Units (Cho et al., 2014) in order to predict the probability of the subsequent events (e.g., item clicks) given a session beginning. Figure 1 shows the general architecture of the network, in which the embedding, the feedforward, and additional GRU layers are optional. In fact, the authors of the method found that a single GRU layer of varying width led to the best performance in their experiments.

The input of the network is formed by a single item, which is one-hot encoded in a vector representing the entire item space, and the output is a

[7] Note that the weighting function is designed to work independently from the similarity function. We rely on the binary session representation for the similarity calculation without considering the order of the items to ensure computational efficiency.

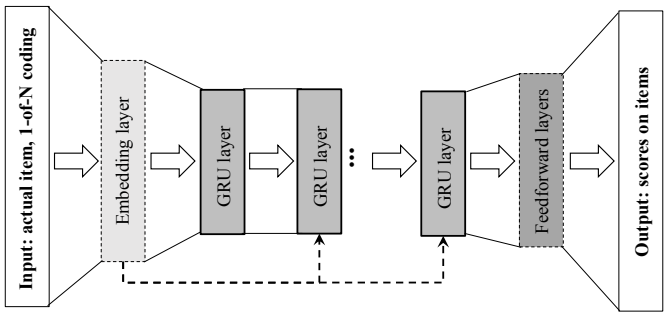


Fig. 1: Architecture of the GRU4REC neural network, adapted from (Hidasi et al., 2016a).

vector of similar shape that should give a ranking distribution for the subsequent item. Inbetween, the standard GRU layer keeps track of a hidden state that encodes the previously occurring items in the same session. Therefore, while training and predicting with the help of this network architecture, the items of a session have to be fed into the network in the correct order and the hidden state of the GRUs has to be reset after a session ends. In terms of the activation functions, the authors found *tanh* and the *sigmoid* function to work best for the GRU and the ranking layer, respectively.

While the usage of RNNs for session-based, or more generally, sequential prediction problems is a natural choice, the particular network architecture, the choice of the loss functions, and the use of session-parallel mini-batches to speed up the training phase are key innovative elements of the approach.

The model can be trained with stochastic gradient descent (SGD) using established optimizations like *ADAM*, *ADADELTA*, *RMSProp*, or *ADAGRAD* (Duchi et al., 2011; Zeiler, 2012; Kingma and Ba, 2014). As common in practice when optimizing deep neural networks, Hidasi et al. train the network in batches. To ensure that the items or events are fed into the network in the correct order, they propose the *session-parallel mini-batch* training scheme, which is illustrated in Figure 2. In the training process, each part of a batch belongs to a specific session in the training data and the network records a separate hidden state for each position. Whenever a session at a position in the batch ends, the corresponding hidden state is reset and the next batch update includes the first event of a new session at that position.

As usual, a number of hyper-parameters can be tuned, including, the learning rate, the layer sizes, a momentum factor, and a drop-out factor to stabilize the network. The choice of the loss function is another key to the quality of the recommendations of GRU4REC. The following loss functions were designed or applied by the authors. In particular the latest function (MAX) proposed by Hidasi and Karatzoglou (2017) led to a significant performance improvement over the previous ones.

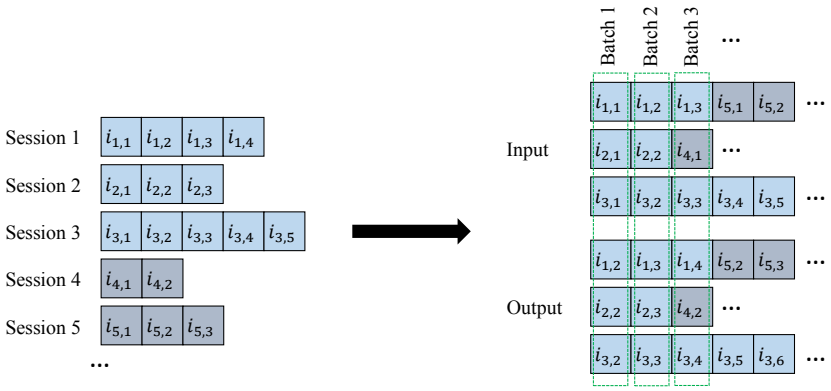


Fig. 2: Illustration of the session-parallel mini-batch scheme of GRU4REC, adapted from (Hidasi et al., 2016a).

– *BPR*: Bayesian Personalized Ranking (BPR), as discussed above, uses a pairwise ranking loss function for the task of creating top-n recommendations. In GRU4REC, a generalized version of this function is applied using the following formula:

$$L_s(\hat{r}_{s,i}, S_N) = -\frac{1}{|S_N|} \cdot \sum_{j \in S_N} log(\sigma(\hat{r}_{s,i} - \hat{r}_{s,j})) \tag{8}$$

In the loss function, the predicted rating $\hat{r}_{s,i}$ for the actual next item i given the current session s is compared to a set of negative samples S_N with the goal of maximizing the difference between them. Here, the sigmoid and logarithm functions are applied to represent the proportion between the ranking of the negative and the positive example.

– *TOP1*: This loss function was introduced by the authors of GRU4REC and can be seen as a regularized approximation of the relative rank of a positive sample $\hat{r}_{s,i}$ and the negative samples S_N:

$$L_s(\hat{r}_{s,i}, S_N) = \frac{1}{|S_N|} \cdot \sum_{j \in S_N} \sigma(\hat{r}_{s,j} - \hat{r}_{s,i}) + \sigma(\hat{r}_{s,j}^2) \tag{9}$$

Here, the proportion is approximated with the sigmoid function, and the regularization term $\sigma(\hat{r}_{s,j}^2)$ is added so that the score of the negative samples is directed to zero.

– *MAX*: In continuation of their work, the authors proposed a generic extension to these two loss functions, where L_s stands for a loss function like *BPR* or *TOP1* defined above:

$$L_{max}(\hat{r}_{s,i}, S_N) = L_s(\hat{r}_{s,i}, \{\max_{j \in S_N} \hat{r}_{s,j}\}) \tag{10}$$

Instead of using a sum of differences between the positive item's rating $\hat{r}_{s,i}$ and the negative samples S_N, only the highest rated negative sample

$\max_{j \in S_N} \hat{r}_{s,j}$ from S_N is used to calculate the loss. As this function has to be differentiable for SGD training, $\max_{j \in S_N}$ is approximated with the *softmax* function. The resulting functions BPR_{max} and $TOP1_{max}$ showed superior performance when compared to the *BPR* and *TOP1* functions (Hidasi and Karatzoglou, 2017).

In our experiments, we used the GRU4REC (v2.0) implementation that the authors shared online. The code is regularly maintained by the authors and includes the implementation of the GRU4REC method, the code of their baseline algorithms, as well as the code for the evaluation procedure proposed in (Hidasi et al., 2016a).

3.4 Factorization-based Methods

As described in Section 2, a number of (hybrid) factorization-based methods were proposed in recent years for *sequential* recommendation problems. We include three existing methods from the literature in our experiments, Factorized Personalized Markov Chains (FPMC) proposed by Rendle et al. (2010), FISM by Kabbur et al. (2013), and FOSSIL by He and McAuley (2016). Generally, these methods aim at predicting the next actions of users, but were not designed for session-based recommendation scenarios with anonymous users. We therefore describe for each method how we applied them to our problem setting. In addition, we propose a novel factorization method called Session-based Matrix Factorization (SMF), which relies on the BPR_{max} and $TOP1_{max}$ loss functions as described above.

3.4.1 Factorized Personalized Markov Chains (FPMC)

The FPMC method was designed for the specific problem of next-basket recommendation. The problem consists of predicting the contents of the next basket of a user, given his or her history of past shopping baskets. By limiting the basket size to one item and by considering the current session as the history of baskets, the method can be directly applied for session-based recommendation problems.

Technically, FPMC combines MC and traditional user-item matrix factorization in a three dimensional tensor factorization approach. As illustrated in Figure 3, the third dimension captures the transition probabilities from one item to another.

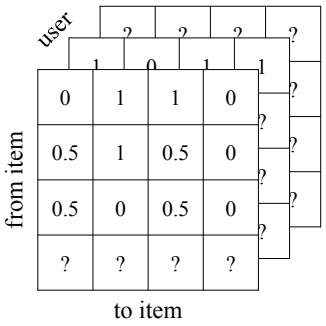


Fig. 3: Personalized transition cube, adapted from (Rendle et al., 2009).

Internally, a special form of the Canonical Tensor Decomposition is used to factor the cube into latent matrices, which can then be used to predict a ranking in the following way:

$$\hat{r}_{u,l,i} = \langle v_u^{U,I}, v_i^{I,U} \rangle + \langle v_i^{I,L}, v_l^{L,I} \rangle + \langle v_u^{U,L}, v_l^{L,U} \rangle \tag{11}$$

where $\hat{r}_{u,l,i}$ is a score for item i with the preferences of user u when he or she previously examined item l. The three-dimensional decomposition results in six latent matrices $v^{X,Y}$ representing the latent factors for dimension X regarding dimension Y, e.g., $v^{U,L}$ are the user latent factors in terms of the previously examined item and $v^{I,L}$ the item latent factors regarding the previously examined item. Accordingly, $v_u^{U,L}$ for example represents the factors for a single user u and $v_i^{I,L}$ the factors for item i, which are combined with the regular dot product ($\langle a, b \rangle$) to calculate the ranking $\hat{r}_{u,l,i}$. Those latent factors are learned using SGD with the pairwise ranking loss function BPR.

In our problem setting, where we have no long-term user histories, each session in the training data corresponds to a user. Once the model is trained, each new session therefore represents a user cold-start situation. To apply the model to our setting, we estimate the session latent vectors as the average of the latent factors of the individual items in the session. This approach was adopted also by Hidasi et al. (2016a) to apply BPR-MF to session-based recommendation scenarios.

*3.4.2 Factored Item Similarity Models (*FISM*)*

This method is based on an item-item factorization, which has the advantage of being directly applicable to our session-based cold-start scenario, where no explicit user representation can be learned. However, FISM does not incorporate sequential item-to-item transitions like FPMC does. Equation 12 shows the prediction function which Kabbur et al. (2013) trained using SGD to predict ratings, e.g., for the movie domain.

$$\hat{r}_{u,i} = b_u + b_i + (n_u^+)^{-\alpha} \sum_{j \in R_u^+} p_j q_i^T \tag{12}$$

Technically, for user u and item i, a score $\hat{r}_{u,i}$ is calculated as the sum of latent vector products $p_j q_i^T$ between item i and the items R_u^+ already rated by the user u. In our scenario, R_u^+ corresponds to the previously inspected items in a session. The terms b_u and b_i are bias terms and n_u^+ specifies the number of ratings by user u, which is combined with a parameter α to normalize the sum of vector products to a certain degree. Instead of using the $RMSE$ as an error metric, we use BPR's pairwise loss function when optimizing the top-n recommendations for the given implicit feedback scenario.

*3.4.3 Factorized Sequential Prediction with Item Similarity Models (*FOSSIL*)*

In this approach, FISM is combined with factorized Markov chains to incorporate sequential information into the model. The model can be described as shown in Equation 13 (from He and McAuley (2016)):

$$\hat{r}_{u,l,i} = \underbrace{\sum_{j \in R_u^+ \setminus \{i\}} p_j q_i^T}_{\text{long-term preferences}} + \overbrace{(w + w_u)}^{\text{personalized weighting}} \cdot \underbrace{n_l m_i^T}_{\text{sequential dynamics}} \tag{13}$$

Again, $\hat{r}_{u,l,i}$ represents a rating for item i given a user u and his or her previously inspected item l. The first term represents the long-term user preferences and corresponds to the FISM model in Equation 12. Using a weighted sum with a global factor w and a personalized factor w_u, the model is extended by a factorized Markov chain to capture the sequential dynamics. In the last term of Equation 13, a latent vector n_l for item l is multiplied with a latent vector m_i for item i to factor in the user-independent probability of item l being followed by item i.

In our scenario, again, the sessions represent the users, R_u corresponds to the current session and BPR is used as the loss function to rank suitable items over negative examples.

*3.4.4 Session-based Matrix Factorization (*SMF*)*

Finally, SMF is a novel factorization-based model that we designed for the specific task of session-based recommendation. Similar to FOSSIL it combines factorized Markov chains with classic matrix factorization. In addition, our method considers the cold-start situation of session-based recommendation scenarios as follows.

In contrast to the traditional factorization-based prediction model $r_{u,i} = p_u q_i^T$, in the SMF method, we replace the latent user vector p_u with a session preference vector s_e, which is computed as an embedding of the current session s:

$$s_e = M_{ST} \cdot s^T \tag{14}$$

Here, the session s is as a binary vector similar to the representation in SKNN (see Section 3.2.2) and M_{ST} is a transformation matrix of size $|I| \cdot |u_s|$, which

reduces the size of the binary session vector (number of unique items $|I|$) to a specific latent vector size $|s_e|$.

Based on the embedded session representation s_e, the prediction function is defined as shown in Equation 15.

$$\hat{r}_{s,l,i} = w_i \cdot (\underbrace{s_e q_i^T + b_{1,i}}_{\text{session preferences}}) + (1 - w_i) \cdot (\overbrace{n_l m_i^T + b_{2,i}}^{\text{sequential dynamics}}) \quad (15)$$

The score $\hat{r}_{s,l,i}$ for a session s with the most recent item l and an item i is computed as a weighted combination of session preferences and sequential dynamics. Here, the session preferences correspond to the long-term user preferences in the traditional matrix factorization model, i.e., the embedded session latent vector s_e for the current session s is multiplied with an item latent vector q_i for item i to compute a relevance score i regarding s. The sequential dynamics are captured exactly as in Equation 13 for FOSSIL using latent representations for the currently inspected item l and item i. Both partial scores are adjusted with a separate bias term $b_{x,i}$ and combined in a weighted sum with the factor w_i dependent on item i.

To train this model, we incorporated some of the concepts from GRU4REC (see Section 3.3). Specifically, we adopted *ADAGRAD* for SGD-based optimization, and used BPR_{max} and $TOP1_{max}$ as loss functions. Furthermore, we integrated two additional concepts (and corresponding hyper-parameters) in the training phase to avoid model over-fitting: a session drop-out factor and a skip-rate. For a drop-out factor of 0.1, for example, each positive entry of the binary session input vector is set to 0 with a probability of 10%. The skip-rate, in contrast, describes how often not the immediate next item in the log data should be used as a positive sample in the training process, but the subsequent one. A skip rate of 0.1 therefore means that in 10 % of the cases the immediate next item is skipped.

4 Experiment Setup

In this section, we describe the details of our algorithm comparison in terms of to the used evaluation protocol, the performance measures, and the evaluation datasets. All source code and pointers to the public datasets are provided online to ensure reproducibility of our research.[8]

4.1 Evaluation Protocol and Performance Measures

The general computational task in session-based recommendation problems is to generate a ranked list of objects that in some form "matches" a given session beginning. What represents a good match, depends on the specific application

[8] `https://www.dropbox.com/sh/dbzmtq4zhzbj5o9/AACldzQWbw-igKjcPTBI6ZPAa?dl=0`

scenario. It could be a set of alternative shopping items in an e-commerce scenario or a continuation of given music listening session.

In offline evaluations for session-based recommendations, researchers often abstract from the underlying purpose of the system (Jannach and Adomavicius, 2016), e.g., if the recommender should help discover something new or find alternatives to a currently inspected item. Instead, the recorded user sessions are typically considered as a "gold standard" for the evaluation. To measure the performance of an algorithm, researchers resort to assessing the capability of an algorithm to predict the withheld entries of a session.

Different approaches are found in the literature to withhold certain entries of a session. In some works, only the last element is hidden (Hariri et al., 2012; Bonnin and Jannach, 2014), some propose to "reveal" the first n elements of a session (Jannach et al., 2015a), while others, finally, evaluate their approaches by iteratively revealing one entry after the other (Hidasi et al., 2016b). We employed the latter iterative revealing scheme in our experiments as it (i) conceptually includes both of the other techniques and (ii) reflects the user journey throughout a session in the best way.

Selection of the Target Item and Accuracy Measures. We measured prediction accuracy in two ways and correspondingly report the results in separate tables.

- First, to establish comparability with existing research, we use an evaluation scheme in which the task is to predict the *immediate next item* given the first n elements. For each session, we iteratively increment n, measure the hit rate (HR) and the Mean Reciprocal Rank (MRR), and finally determine the average HR and MRR for all sessions for the different list lengths, as done by Hidasi et al. (2016b).
- Second, instead of focusing only on the next item, we made a measurement where we considered *all subsequent* elements for the given session beginning, because all of them might be relevant to the user. In this scheme, we used the standard information retrieval measures precision and recall at defined list lengths. The number of given elements of the session is also iteratively incremented as in the previously described evaluation scheme.

Sessionization strategies. Different strategies exist in the literature to split the user activity logs into sessions. In some of the public datasets used in our evaluation, the activity logs were already split up into sessions, i.e., each log entry was assigned a unique session ID (*RSC15*, *Zalando*). For other datasets (*RETAILR*, *NOWPLAYING*, *30MUSIC*, *CLEF*), we used a common heuristic-based approach and considered a session as over after a defined user idle time, e.g., 30 minutes of user inactivity (Cooley et al., 1999). For the TMALL dataset, where the timetamps for the recorded events were only available at the granularity of a day, we considered all events of one day as belonging to one session. Finally, for the two playlists dataset (*AOTM*, *8TRACKS*), we considered all elements of a playlist to be part of a session.

Training and Test Splits, Repeated Subsampling. Hidasi et al. (2016b) used one single training-test split. In the case of an e-commerce dataset, the data was split in a way that the sessions of all six months except those of the very last day of the entire dataset were placed in the training set. The last day was used for testing. We report the results of applying this evaluation scheme to ensure comparability, e.g., with respect to the results obtained for the e-commerce dataset that was used in their experiments.

Since such single-split setups have their limitations, we focus our discussion on the results that were obtained when applying a *sliding-window* protocol, where we split the data into 5 slices of equal size in days. For most e-commerce data, for example, we used the data of about one month for training and the subsequent data (e.g., of one day) for testing (see Section 4.2 for the dataset specific configurations). This allows us to make multiple measurements with different test sets. We then evaluate the performance for each of these data samples and report the average of the performance results for all slices. This latter protocol helps us reduce the danger that the observed outcomes are the results of one particular train-test configuration.[9]

For the playlist datasets *8TRACKS* and *AOTM* no timestamp information is available. For these datasets we therefore applied a standard cross-validation procedure, where elements are randomly assigned to the training and test sets. We did not use such a time-agnostic data splitting procedure for the e-commerce and news datasets for different reasons. First, as the results will show, there are strong temporal effects that should be considered in the recommendation process. Second, in these domains, the set of items is not static and in particular in the news domain new items appear constantly. Randomly splitting the sessions would then potentially result in the effect that future interactions with not-yet-existing items would be considered in the training phase.

Additional Quality Factors. Since accuracy is not the only relevant quality factor in practice, we made the following additional measurements, as was done by Jannach and Ludewig (2017).

- *Coverage:* We report how many different items ever appear in the top-k recommendations. This measure represents a form of catalog coverage, which is sometimes referred to as *aggregate diversity* (Adomavicius and Kwon, 2012).
- *Popularity bias:* High accuracy values can, depending on the measurement method, correlate with the tendency of an algorithm to recommend mostly popular items (Jannach et al., 2015b). To assess the popularity tendencies of the tested algorithms, we report the *average popularity score* for the elements of the top-k recommendations of each algorithm. This average

[9] To ensure that the smaller size of those splits does not negatively affect the performance of the model-based approaches, we tested the single-split configurations as well on all datasets. The obtained results are mostly in line with those obtained with the sliding-window protocol and shown in Appendix D.

score is the mean of the *individual popularity scores* of each recommended item. We compute these scores based on the training set by counting how often each item appears in one of the training sessions and by then applying min-max normalization to obtain a score between 0 and 1.
- *Cold start:* Some methods might only be effective when a significant amount of training data is available. We, therefore, report the results of measurements where we artificially removed parts of the (older) training data to simulate such situations.
- *Scalability:* Training modern machine learning methods can be computationally challenging, and obtaining good results may furthermore require extensive parameter tuning. We, therefore, report the times that the algorithms needed to train the models and to make predictions at runtime. In addition, we report the memory requirements of the algorithms.

By reporting quality factors *coverage* and *popularity bias* our aim is to emphasize that different recommendation strategies can lead to quite different recommendations, even if they are similar in terms of the prediction accuracy, see also (Jannach et al., 2015b). Such multi-metric evaluation approaches should also help practitioners to better understand the potential side effects of the recommenders, e.g., reduced average sales diversity and additionally increased sales of top-sellers (Lee and Hosanagar, 2014). It remains however difficult to aggregate the individual performance factors into one single score, as the relative importance of the factors can depend not only on the application domain, but also on the specific business model of the provider.

Parameter Optimization. Some of the algorithms that we tested require extensive (hyper-)parameter tuning including SMF and GRU4REC. Thus, we systematically optimized the parameters for those algorithms for each dataset. Due to the computational complexity of the methods, we restricted the layer size for GRU4REC as well as the number of latent factors for SMF to 100 and used a randomized search method with 100 iterations for the remaining parameters as described by Hidasi and Karatzoglou (2017). In each iteration, the learning rate, the drop-out factor, the momentum, and the loss function were determined in a randomized process to find the maximum hit rate for a list length of 20. All optimizations were performed on special validation splits, which were created by splitting a training set into a validation training and test set. For the simpler S-KNN-based approaches, we used the same validation sets to manually adjust the number of neighbors and samples when applying cosine similarity as the distance measure (except for V-SKNN). The final parameters for each method and dataset are provided in Appendix A.

4.2 Datasets

We made measurements for datasets from three different domains: e-commerce, music, and news.

Table 1: Characteristics of the e-commerce datasets. The values are averaged over all five non-overlapping splits for each dataset, except for *RSC15-S*, where we only use one train-test split.

Dataset	RSC15-S	RSC15	TMALL	RETAILR	ZALANDO
Actions	31.71M	5.43M	13.42M	212,182	4.54M
Sessions	7.98M	1.38M	1.77M	59,962	365,126
Items	37,483	28,582	425,348	31,968	189,328
Timespan in Days	182	31	91	27	91
Actions per Session	3.97	3.95	7.56	3.54	12.43
Unique Items per Session	3.17	3.17	5.56	2.56	8.39
Actions per Day	174,222	175,063	149,096	7,858	50,410
Sessions per Day	43,854	44,358	19,719	2,220	4056

E-Commerce Datasets. We used the following four e-commerce datasets.

- *RSC15.* This is one of the datasets that was used in (Hidasi et al., 2016a) and their later works. It was published in the context of the ACM RecSys 2015 Challenge and contains recorded click sequences (item views, purchases) for a period of six months. We use the label *RSC15-S* to denote the dataset and measurement where only one single train-test split is used. For RSC15, each split consists of 30 days of training and 1 day of test data.
- *TMALL.* This dataset was published in the context of the TMall competition and contains interaction logs of the tmall.com website for one year. For TMALL, each split consists of 90 days of training and 1 day of test data.
- *RETAILR.* The e-commerce personalization company *retailrocket* published this dataset covering six month of user browsing activities, also in the context of a competition. For RETAILR, each split consists of 25 days of training and 2 days of test data.
- *ZALANDO.* The final dataset is non-public and was shared with us by the fashion retailer Zalando. It contains user logs of their shopping platform for a period of one year. In our evaluation, we only considered the item *view* events as was done for the other e-commerce datasets. For ZALANDO, each split consists of 90 days of training and 1 day of test data.

Table 1 shows an overview of the characteristics of the e-commerce datasets. Except for the *RSC15-S* dataset, which we include to make our evaluation comparable with previous works (Hidasi and Karatzoglou, 2017; Jannach and Ludewig, 2017), we report the average values after creating five data splits as described above.

Media Datasets: Music and News. As in (Jannach and Ludewig, 2017), we use the music domain as an alternative area to evaluate session-based recommendation algorithms, because music is commonly consumed within listening sessions in sequential order. We use the same datasets that were used in (Jannach and Ludewig, 2017), which consist of two sets of *listening logs* and two

datasets of user-created *playlists*. In addition, we made measurements using a dataset from the news recommendation domain.

We in particular consider the news domain because it has certain distinct characteristics (Karimi et al., 2018). First, constantly new items become available for recommendation (Das et al., 2007; Liu et al., 2010). At the same time, items can also quickly become outdated. Second, previous research indicates that short-term popularity trends can be important for the success of a recommender (Ludmann, 2017). The experiments based on this dataset should therefore provide an indicator if the general insights obtained from other domains generalize to a domain with very specific characteristics.

- *8TRACKS and AOTM:* These dataset include playlists created by music enthusiasts. The *AOTM* dataset was collected from the Art-of-the-Mix platform and is publicly available (McFee and Lanckriet, 2012). The non-public *8TRACKS)* dataset was shared with us by the 8tracks.com music platform. For all music datasets, each split consists of 90 days of training and 5 days of test data.
- *30MUSIC and NOWPLAYING:* The *30MUSIC* dataset contains listening histories of the last.fm music platform and was published by Turrin et al. (2015). The *NOWPLAYING* dataset was created from music-related tweets, where users posted which tracks they were currently listening (Zangerle et al., 2014).
- *CLEF:* The dataset was made available to participants of the 2017 CLEF NewsREEL challenge.[10] It consists of a stream of user actions (e.g., article reads) and article publication events, which were collected by the company *plista* for several publishers. In our evaluation we only considered the article read events. We used the data of the publisher with the largest amount of recorded interactions (the popular German sports news portal Sport1[11]). For CLEF, each split consists of 5 days of training and 1 days of test data.

The statistics for the datasets from the media (music and news) domain are given in Table 2.

5 Results

5.1 E-Commerce Datasets

Table 3 shows the MRR and Hit Rate results at a recommendation list length 20 for the four tested e-commerce datasets. In addition, we report the results when applying the standard measures precision and recall when considering *all* hidden elements in the rest of the session as described above (see Table 4). Finally, we also report coverage and popularity statistics for each algorithm.

[10] `http://www.clef-newsreel.org/`

[11] `https://www.sport1.de/`

Table 2: Characteristics of the music and news datasets. The values are again averaged over all five non-overlapping splits.

	8TRACKS	30MUSIC	AOTM	NOWPLAYING	CLEF
Actions	1.50M	638,933	306,830	271,177	5.54M
Sessions	132,453	37,333	21,888	27,005	1.64M
Items	376,422	210,633	91,166	75,169	742
Timespan in Days	95	95	95	95	6
Actions per Session	11.32	17.11	14.02	10.04	3.37
Items per Session	11.31	14.47	14.01	9.38	3.17
Actions per Day	16,663	7,099	3,409	3,013	923,414
Sessions per Day	1,472	415	243	300	274,074

5.1.1 Accuracy Measures

The results when the task is to predict the *immediate next* element in a session (as done in (Hidasi and Karatzoglou, 2017; Jannach and Ludewig, 2017)) are shown in Tables 3a to 3e. The following observations in terms of the hit rate and the MRR can be made.[12]

- The lowest accuracy values are almost consistently achieved across all datasets by the family of Factorized Markov Chain approaches (FISM, FPMC and FOSSIL) and the session-aware BPR-MF variant. BPR-MF in fact often exhibits the best performance among these methods even though it was not designed for sequential recommendation problems. In two cases, however, the session-based BPR-MF variant led to very competitive results when the measurement was taken at a recommendation list length of 1, although at the potential price of a high popularity bias and low coverage. Apart from this phenomenon, our results indicate that the methods that were designed under the assumption of longer-term and richer user profiles are often not particularly well suited for the specifics of session-based recommendation problems.
- The simple pairwise association methods (AR and SR) mostly occupy the middle places in our comparison. In most cases, it is preferable to consider the available sequentiality information (SR). Only for the *TMALL* dataset, where the transactions of an entire day are considered as a session[13], and for *RETAILR*, the sequence-agnostic AR method is slightly better in terms of the hit rate. In terms of the overall ranking, the trivial SR method is, to some surprise, among the *top-performing* methods for two of the datasets in terms of the MRR, with good results also for the hit rate. The MC method finally, is usually placed somewhere in the middle of the ranking. Similar to the SR method, it is very strong in terms of the MRR for two of the datasets.

[12] We provide additional results that were obtained for measurements taken at multiple list lengths in Appendix B.

[13] In the dataset, timestamps are only available at the granularity of days.

Table 3: Hit rate (HR), Mean reciprocal rank (MRR), catalog coverage (COV), and the average popularity (POP) for a list length of 20 obtained for the e-commerce datasets. The table rows are ordered by MRR@20. The best values are highlighted in each column and, in case of accuracy measures, marked with a star when the difference w.r.t. to the second-best performing method was statistically significant.

(a) RSC15

Metrics	MRR@20	HR@20	COV@20	POP@20
GRU4REC	**0.308**	***0.683**	0.504	0.054
SR	0.304	0.653	0.668	0.072
SMF	0.302	0.666	0.565	0.055
MC	0.300	0.642	0.645	0.070
AR	0.289	0.636	0.630	0.093
V-SKNN	0.283	0.653	0.619	0.079
S-SKNN	0.272	0.602	0.655	0.072
SF-SKNN	0.270	0.589	0.619	0.066
S-KNN	0.266	0.621	0.634	0.073
IKNN	0.208	0.486	0.755	0.041
FPMC	0.201	0.363	**0.975**	0.055
BPR-MF	0.176	0.235	0.911	0.088
FISM	0.115	0.162	0.974	**0.008**
FOSSIL	0.062	0.190	0.917	0.048

(b) TMALL

Metrics	MRR@20	HR@20	COV@20	POP@20
S-SKNN	***0.185**	0.387	0.467	0.025
S-KNN	0.182	***0.404**	0.381	0.026
V-SKNN	0.179	0.373	0.464	0.024
BPR-MF	0.159	0.204	0.723	0.057
SF-SKNN	0.136	0.216	0.436	0.018
GRU4REC	0.129	0.277	0.151	0.035
AR	0.129	0.262	0.509	0.021
SR	0.128	0.242	0.569	0.021
SMF	0.121	0.261	0.261	0.036
MC	0.116	0.200	0.498	0.019
FPMC	0.101	0.119	**0.880**	0.005
IKNN	0.051	0.150	0.728	0.007
FISM	0.024	0.037	0.752	**0.003**
FOSSIL	0.001	0.004	0.598	0.016

(c) RETAILR

Metrics	MRR@20	HR@20	COV@20	POP@20
S-SKNN	***0.345**	***0.591**	0.596	0.056
V-SKNN	0.338	0.573	0.575	0.060
S-KNN	0.337	0.583	0.566	0.058
BPR-MF	0.303	0.357	0.824	0.060
FPMC	0.273	0.320	**0.929**	0.022
SF-SKNN	0.260	0.358	0.403	0.035
SR	0.245	0.419	0.524	0.042
GRU4REC	0.243	0.480	0.602	0.060
AR	0.241	0.439	0.544	0.053
MC	0.230	0.359	0.411	0.035
SMF	0.225	0.459	0.449	0.085
IKNN	0.107	0.240	0.584	0.033
FISM	0.075	0.132	0.848	**0.018**
FOSSIL	0.022	0.058	0.753	0.127

(d) ZALANDO

Metrics	MRR@20	HR@20	COV@20	POP@20
SR	**0.304**	0.483	0.586	0.061
MC	0.303	0.455	0.513	0.060
IKNN	0.275	0.405	0.714	0.037
GRU4REC	0.267	0.468	0.304	0.101
SMF	0.267	0.447	0.362	0.107
AR	0.258	0.467	0.467	0.089
SF-SKNN	0.249	0.438	0.432	0.057
V-SKNN	0.233	***0.521**	0.432	0.096
S-SKNN	0.219	0.499	0.435	0.087
S-KNN	0.172	0.456	0.309	0.093
BPR-MF	0.104	0.162	0.609	0.058
FPMC	0.051	0.075	**0.812**	0.021
FISM	0.004	0.011	0.624	**0.020**
FOSSIL	0.002	0.005	0.671	0.034

(e) RSC15-S

Dataset	RSC15	
Metric	MRR@20	HR@20
GRU4REC	**0.312**	**0.719**
SMF	0.309	0.713
SR	0.308	0.690
V-SKNN	0.274	0.675
SKNN	0.250	0.641

Table 4: Precision (P@20) and Recall (R@20) for the e-commerce datasets. The rows are ordered by the P@20 values for the *TMALL* data set, which led to a relatively consistent ranking of the algorithms.

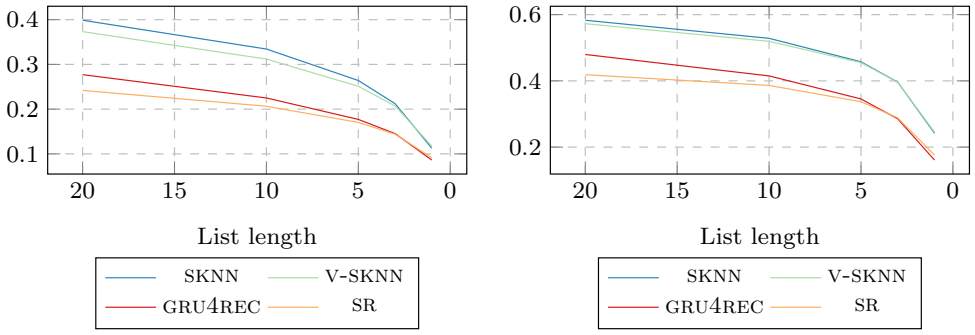
Dataset	RSC15		TMALL		ROCKET		ZALANDO	
Metric	P@20	R@20	P@20	R@20	P@20	R@20	P@20	R@20
SKNN	0.086	0.464	**0.095**	**0.312**	**0.056**	**0.478**	0.074	0.202
V-SKNN	**0.092**	0.494	0.088	0.291	0.055	0.462	**0.076**	**0.207**
SMF	0.092	**0.501**	0.068	0.230	0.047	0.397	0.062	0.175
GRU4REC	0.085	0.470	0.068	0.233	0.046	0.400	0.065	0.181
SR	0.089	0.488	0.052	0.193	0.038	0.342	0.060	0.174

- The performance of the newly-proposed SMF method is very strong for the *RSC15* and *RSC15-S* dataset and in the middle ranges for the other datasets. The SMF method consistently outperforms the factorization-based methods from the literature, apparently due to the embedding of the current user session.
- GRU4REC is consistently among the top five algorithms in this comparison in terms of the hit rate and exhibits competitive performance results also with respect to the MRR. The method is outperforming all other methods on the *RSC15(-S)* datasets in terms of the hitrate and is competitive w.r.t. the MRR, where the differences between the top-performing methods are tiny. On the other datasets, the accuracy results of GRU4REC are, however, often significantly lower than those of the best-performing methods.[14]
- For each of the datasets, one of the proposed neighborhood-based methods was usually the winner in terms of the hit rate and the MRR (except for *RSC15(-S)* and the MRR on *ZALANDO*). Using one of the variants that considers sequentiality information is usually favorable, except for the case of the *TMALL* dataset. The most consistent performance of the neighborhood-based methods is achieved with the V-SKNN method which uses a specific sequence-aware similarity measure that gives more weight to the most recent interactions. Generally, the results suggest that there is even room for further improvement in the context of the neighborhood-based methods. In the experiments reported in this work, we could, for example, observe that using a slightly different similarity measure already led to substantial performance improvements for some of the datasets.

Precision and Recall for the Remaining Session. The ranking of the *best-performing algorithms* when evaluating *all* subsequent elements of a session (not only the immediate next click) and measuring precision and recall is given in Table 4. We report the detailed results for all algorithms for all measurements in the appendix.

The obtained results are mostly in line with the previously reported observations. The best performance is achieved by the neighborhood-based methods, with V-SKNN working very well across all datasets. Differently from the previous measurement, GRU4REC shows a lower performance for the *RSC-15* dataset than the other methods. This is probably due to the fact that GRU4REC is optimized to predict the *immediate* next action. Generally, which type of accuracy measurement—focusing on the prediction of the immediate next element or considering the prediction of any item that is relevant in the session as a success—is more appropriate, depends on the application domain. Our results show that the kNN-based methods are successful in both forms, i.e., they are often good at predicting the next element while, at the same time, they many times include *more* items that are relevant for the given session than, e.g., GRU4REC.

[14] We applied the Wilcoxon signed-rank test ($\alpha = 0.05$) to determine the significance of differences between the two best performing approaches for each dataset.



(a) TMALL (b) RETAILR

Fig. 4: Hit rate (HR) for two of the e-commerce datasets when reducing the recommendation list length from 20 to 1.

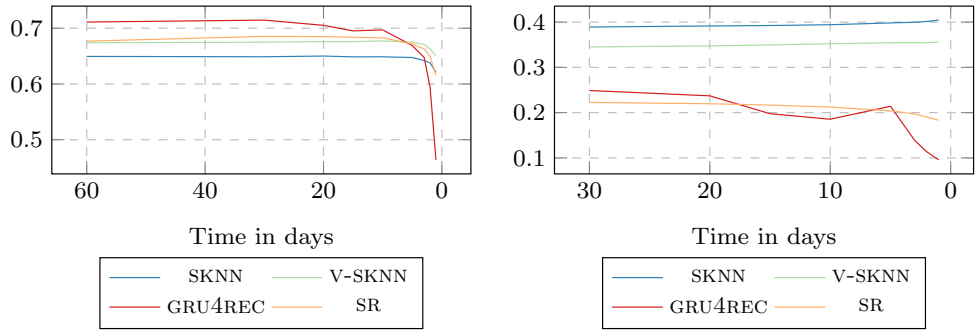
Impact of Different List Lengths To see if the recommendation list length at which the measurement is taken has an influence on the algorithm ranking, we varied the length from 20 to 1. Figure 4a and Figure 4b show how the best algorithms perform for the *TMALL* and *RETAILR* datasets when different list lengths are used in the evaluation. The results show that the ranking of the algorithms can in fact be affected by the change of the list length.

Specifically, the differences between the nearest-neighbor methods and the GRU4REC and SR methods becomes gradually smaller for shorter list lengths. This is not too surprising because both GRU4REC and SR focus on the prediction of the immediate next action and often lead to better performance values in terms of the MRR. Since the particular evaluation protocol here also only focuses on the correct prediction of the next item, the effect might however be overemphasized due to the specific measurement method. An interesting observation is that at list length 1, BPR-MF and to some extent the FPMC method lead to the best results for some e-commerce datasets. In the case of BPR-MF, this however comes at the price of a high popularity tendency of the algorithm and a comparably low coverage (see Table 15 in Appendix B).

5.1.2 Cold-Start and Sparsity Effects

Previous experiments on the *RSC15* dataset revealed that discarding major parts of the older data has no strong impact on the prediction accuracy, at least in the e-commerce domain (Jannach and Ludewig, 2017). We therefore made additional experiments to analyze the effects in more depth. Figure 5a and Figure 5b show the results of this simulation for two of the e-commerce datasets.

The results for the *RSC15-S* (single-split) dataset (Figure 5a) are in line with what was previously reported in (Jannach and Ludewig, 2017). In the e-commerce domain, the user behavior seems to be strongly influenced by recent sales trends, an effect that was also reported in (Jannach et al., 2017b). Discarding most of the historical data has almost no influence on the resulting hit



(a) RSC15-S (b) TMALL

Fig. 5: HR@20 for two e-commerce datasets when artificially reducing the size of the training set from 60 days to 1 day.

rates. This behavior is similar for all compared algorithms. Only in the extreme case when only the data of the last few days is considered, the performance of the algorithms degrades. A similar observation can be made for the *TMALL* dataset. Generally, the observations also explain why the recency-based neighborhood sampling approach implemented in the kNN methods does not have a strong negative effect on the accuracy. In fact, focusing on the most recent sessions when looking for similar neighbors has shown to have a positive effect in (Jannach and Ludewig, 2017), when compared to a random neighborhood selection scheme.

Considering other Types of Events In the reported experiments, the models were trained with past item view events and we also predicted the next view event for a given session beginning. This choice was made to make our work comparable with previous research. Additional types of events (e.g., "add-to-wishlist", "add-to-cart") can easily incorporated as positive preference signals into the investigated algorithms. How to weight the different types of signals nd how to interpret signals like "remove-from-wishlist" is an area for future research.

Depending on the domain, also different types of events might be in the focus as well in the prediction phase. In the experiments reported here, we predict item views. In our previous research on the topic (Jannach and Ludewig, 2017), we also made experiments in which we focused on the prediction of purchase events. In these experiments, the ranking of the algorithms was similar for the item prediction and the purchase prediction tasks. However, some previous research suggests that view-based collaborative filtering algorithms lead to sometimes quite different recommendations than purchase-based ones and also differ in their effectiveness (Lee and Hosanagar, 2014). In general, the choice of the prediction target should therefore be made with the goal of the recommender in mind, e.g., increase user attention and click-through-rates vs. increasing sales, see, e.g., (Jannach and Hegelich, 2009).

5.1.3 Coverage and Popularity Bias

The results listed in Table 3a to Table 3d show that in terms of the coverage (or: aggregate diversity), the factorization-based methods consistently lead to the highest values, i.e., they place the largest number of different items into the top-n lists of the users. GRU4REC represents in all datasets, except *RETAILR*, the other extreme and seems to focus its recommendations on a comparably narrow range of items. In particular in the case of the *TMALL* dataset, the coverage of the item space of GRU4REC is as low as 0.15, i.e., the top-20 recommendations for all given sessions in the test set cover only 15 % of the available items. To what extent low coverage is undesired, again depends on the specific application domain.

Not many consistent patterns can be identified with regard to the popularity biases of the different algorithms. BPR-MF, as was previously discussed by Jannach et al. (2015b), has a comparably strong tendency to focus on generally popular items. Our newly proposed SMF method exhibits a similar tendency across all datasets. The FPMC method usually represents the other end of the spectrum. The tendency of the many of the other algorithms to recommend popular items seems to strongly depend on the dataset characteristics. According to our previous work (Jannach and Ludewig, 2017), the basic SKNN method tends to recommend slightly more popular items than GRU4REC. In this new series of measurements, this is, however, not consistently the case across the datasets.

5.2 Media Datasets

Table 5, Table 6, and Table 7 show the results for the music and news domains, respectively.

Accuracy The accuracy results generally exhibit similar patterns as the results obtained for the e-commerce datasets. For these datasets, however, the winning strategy more strongly depends on the chosen measure. When the MRR is used as a performance measure, often the trivial baselines SR or AR lead to the best results. In terms of the hitrate, in contrast, usually one of the nearest neighbor methods again performs best.

With respect to the MRR measure also GRU4REC exhibited very competitive performance, except for the *8TRACKS* and *AOTM* datasets, where the highest MRR values were achieved with the AR and the SKNN method. Looking at the playlist datasets (*8TRACKS* and *AOTM*), the comparably good results of the sequence-agnostic AR and SKNN strategy indicate that the ordering of the tracks is not too important for the playlist creators. Among the neighborhood-based methods, V-SKNN was again consistently among the top-performing methods. When looking at the standard precision and recall measurements for the five best-performing approaches in Table 6, we can see

Table 5: Hit rate (HR), Mean reciprocal rank (MRR), catalog coverage (COV), and the average popularity (POP) for a list length of 20 tested on the music datasets (Table 5a to 5d). The tables show the top ten algorithms ordered by MRR@20. Again, the best results are highlighted and significant differences are marked with a star.

(a) NOWPLAYING

Metrics	MRR@20	HR@20	COV@20	POP@20
SR	**0.105**	0.203	0.466	**0.025**
GRU4REC	0.102	0.197	0.433	0.052
MC	0.097	0.158	0.294	0.028
SF-SKNN	0.095	0.165	0.277	0.031
SMF	0.088	0.183	0.242	0.092
V-SKNN	0.078	0.255	0.428	0.064
S-SKNN	0.078	***0.262**	0.415	0.062
AR	0.071	0.208	0.453	0.051
S-KNN	0.069	0.243	0.301	0.069
IKNN	0.057	0.182	**0.580**	0.029

(b) 8TRACKS

Metrics	MRR@20	HR@20	COV@20	POP@20
AR	***0.0071**	0.0255	0.4529	0.0912
SMF	0.0064	0.0230	0.1527	0.0864
SR	0.0063	0.0170	0.4967	0.0531
SF-SKNN	0.0063	0.0118	0.3049	0.0362
V-SKNN	0.0057	0.0352	0.4080	0.1194
S-KNN	0.0053	***0.0375**	0.2430	0.1079
IKNN	0.0050	0.0176	**0.6956**	**0.0245**
GRU4REC	0.0050	0.0189	0.0692	0.1222
S-SKNN	0.0047	0.0293	0.4509	0.0806
MC	0.0046	0.0098	0.3496	0.0320

(c) 30MUSIC

Metrics	MRR@20	HR@20	COV@20	POP@20
SR	***0.238**	0.332	0.389	0.023
MC	0.232	0.284	0.204	**0.021**
GRU4REC	0.226	0.326	0.345	0.056
SF-SKNN	0.208	0.286	0.185	0.022
SMF	0.178	0.284	0.151	0.105
V-SKNN	0.110	0.382	0.317	0.054
IKNN	0.109	0.297	**0.460**	0.023
S-SKNN	0.108	***0.386**	0.293	0.052
AR	0.096	0.309	0.352	0.039
S-KNN	0.090	0.344	0.191	0.057

(d) AOTM

Metrics	MRR@20	HR@20	COV@20	POP@20
SMF	**0.0111**	0.0297	0.2456	0.1998
SF-SKNN	0.0110	0.0144	0.3558	0.0508
SR	0.0076	0.0195	0.5864	0.0533
GRU4REC	0.0071	0.0156	0.4652	0.1151
MC	0.0063	0.0132	0.3803	0.0497
AR	0.0059	0.0233	0.5531	0.1049
V-SKNN	0.0054	0.0377	0.5362	0.1397
S-SKNN	0.0054	0.0397	0.5356	0.1289
S-KNN	0.0053	***0.0429**	0.2802	0.1677
IKNN	0.0049	0.0186	**0.7879**	**0.0472**

Table 6: Precision (P@20) and Recall (R@20) for the music datasets. The results are ordered by P@20 for *8TRACKS*, which represents the largest music dataset in our evaluation.

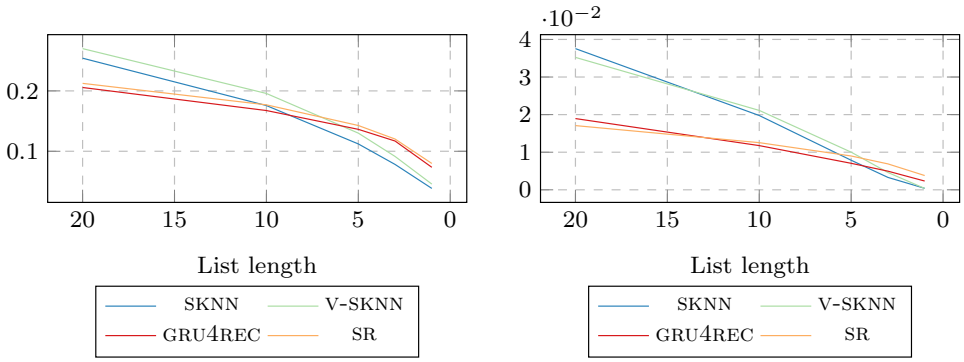
Dataset	LFM		8TRACKS		30MUSIC		AOTM	
Metric	P@20	R@20	P@20	R@20	P@20	R@20	P@20	R@20
V-SKNN	**0.0717**	**0.1909**	**0.0122**	0.0308	**0.1094**	**0.2321**	0.0133	0.0361
SKNN	0.0680	0.1824	0.0117	**0.0313**	0.1035	0.2140	**0.0155**	**0.0440**
SMF	0.0499	0.1453	0.0086	0.0218	0.0746	0.1655	0.0084	0.0259
SR	0.0501	0.1465	0.0055	0.0140	0.0878	0.2010	0.0053	0.0146
GRU4REC	0.0272	0.0810	0.0037	0.0095	0.0404	0.0988	0.0010	0.0027

that V-SKNN is the winning strategy across all datasets and that GRU4REC is again less effective for this particular measurement.

Finally, looking at the *news* domain, the average results shown in Table 7 in general confirm the trends observed for the other datasets. The V-SKNN method is top-performing on almost all measures. GRU4REC also works comparably well on this dataset, especially on the precision and recall measures. Again, however, we can also observe a comparably low level of coverage and

Table 7: Hit rate (HR), Mean reciprocal rank (MRR), Precision (P), Recall (R), item coverage (COV), and average popularity (POP) results for a list length of 20 on the *CLEF* dataset (ordered by MRR@20).

Metrics	MRR@20	HR@20	COV@20	POP@20	P@20	R@20
SMF	**0.234**	0.706	0.650	0.083	0.062	0.527
V-SKNN	0.224	0.776	0.621	**0.083**	0.068	0.584
SR	0.223	0.672	**0.655**	0.093	0.058	0.502
GRU4REC	0.220	0.568	0.174	0.094	**0.072**	**0.626**
S-KNN	0.219	**0.778**	0.613	0.084	0.066	0.577



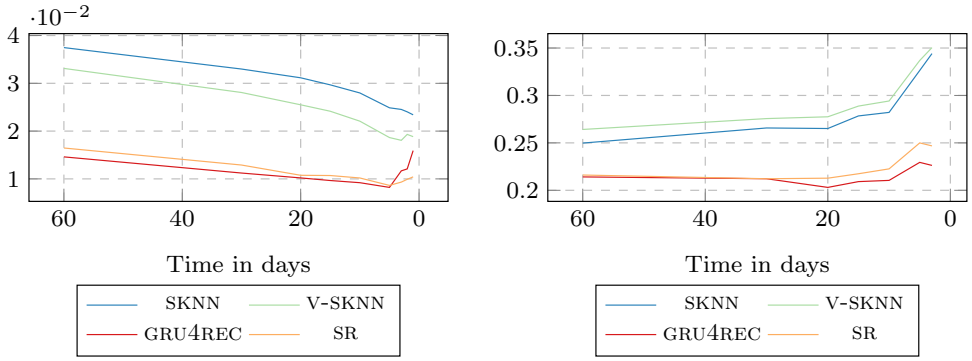
(a) NOWPLAYING (b) 8TRACKS

Fig. 6: Hit rate (HR) for two of the music datasets when reducing the result list length from 20 to 1.

a comparably high tendency to recommend popular items. In contrast to all other domains and datasets however, when looking at the results of the individual splits for the *CLEF* dataset, we could observe that those are subject to large fluctuations. Depending on the day that was chosen for testing, the ranking of the algorithms in terms of the accuracy measures changes drastically, which we could not observe for any other dataset.

As mentioned in Section 4, we conducted additional single split experiments to ensure that the reduced amount of training data in the sliding window protocol does not affect the performance of the model-based approaches. The single-split results in Appendix D reveal GRU4REC as the best-performing approach for this particular dataset, which was also the case for two of the individual splits. Thus, even though such large fluctuations did only occur in the news domain, this is an indicator that applying a single-split evaluation protocol can easily lead to "random" and misleading results.

The effects when considering different list lengths for two of the datasets is shown in Figure 6a (*NOWPLAYING*) and Figure 6b (*8TRACKS*). In contrast to the e-commerce datasets, the relative ranking of the algorithms even changes when the list lengths become shorter. For both datasets, the GRU4REC method and the very simple AR and SR methods, respectively, are better in terms of the hit rate when it comes to very short list lengths. Considering the good results



(a) 8TRACKS (b) NOWPLAYING

Fig. 7: HR@20 for two music datasets when incrementally reducing the size of the training set to 1 day.

for the MRR for these methods (Table 5a and Table 5b), this was expected. Again, the good performance of certain methods can be explained by the fact that these methods are optimized to predict the immediate next item of a given session.

Cold-Start and Sparsity Effects An interesting effect can be observed when older data is discarded to simulate sparsity effects. Figure 7a and Figure 7b show the results for the *8TRACKS* and *NOWPLAYING* datasets, respectively.[15] While for the *8TRACKS* playlist dataset the accuracy values more or less consistently decrease when older data is discarded, we can observe an *increase* in accuracy for the *NOWPLAYING* dataset. Remember that this dataset is based on the analysis of user posts on Twitter about their current listening behavior. Obtaining the highest accuracy values when only considering the very last days means that this dataset is strongly dominated by short-term popularity trends and that the recommendation of older, non-trending tracks is detrimental to the accuracy results.

Coverage and Popularity Bias In terms of coverage (see Table 5), the findings for datasets from the media domain are also mostly in line with those for the e-commerce datasets. The ranking of the algorithms varies largely across the datasets. The differences are, however, often less pronounced. Regarding the popularity tendency of the algorithms, methods that are based on pairwise sequences (SR and MC) in most cases lead to the recommendation of lesser known items, while nearest-neighbor-based techniques quite often focus on the recommendation of comparably popular objects.

[15] The other media datasets did not exhibit any notable particularities.

Table 8: Overview of Computation Times and Memory Requirements for the *RSC15-S* dataset and the first split of the *8TRACKS* dataset, ordered in terms of required training times for the *RSC15-S* dataset.

Dataset	RSC15-S			8TRACKS		
Technique/Metric	Train. (min)	Pred. (ms)	Mem. (MB)	Train.	Pred.	Mem.
MC	0.77	3.34	38	0.05	14.71	144
S-KNN	1.24	33.05	6051	0.04	52.96	353
S-SKNN	1.26	30.26	6051	0.05	51.01	353
V-SKNN	1.30	32.67	6051	0.04	52.43	353
SF-SKNN	1.72	29.82	6254	0.06	18.82	493
SR	2.41	3.14	54	0.18	15.88	284
AR	3.00	3.36	40	0.28	15.79	257
FISM	356.84	8.40	4937	35.07	60.72	387
GRU4REC (GPU)	385.35	7.43	59	19.08	58.08	588
BPR-MF	392.60	8.37	8009	42.69	65.15	771
SMF (GPU)	446.66	14.02	1640	77.50	43.36	1805
FPMC	469.39	9.08	6786	60.92	71.58	1302
FOSSIL	499.19	10.56	4987	50.99	64.91	582

5.3 Computational Complexity & Memory Usage

The methods included in our comparison vary largely in terms of the computational complexity and their memory requirements. Since neighborhood-based methods do not scale well when applied in a naive manner, we used implementation variants that rely on neighborhood sampling and specific in-memory data structures. The comparison of SKNN method and GRU4REC in (Jannach and Ludewig, 2017) showed that, with such an implementation, recommendations can be quickly computed at prediction time with nearest neighbor methods, even though the prediction performance of model-based techniques like GRU4REC could not be achieved.

To enable comparability with previous research (Jannach and Ludewig, 2017), we report the running times and memory demands for the single-split *RSC15-S* dataset, which is also the largest one in terms of the recorded user actions. Additionally, we include the *8TRACKS* dataset, which is rather small compared to *RSC15-S* in terms of the number of events, but has the largest product catalog of all datasets. Table 8 shows the times required for training the model (if applicable), the time needed to compute a recommendation at prediction time, and the memory requirements for the internal data structures. The reported results were obtained when using an Intel Core i7 4790K processor with 32GB of DDR3-1600 memory and a Nvidia GeForce GTX 960 graphics card with 2GB of memory. The following observations can be made.

Running Times The simple methods in our comparative evaluation need from less than one to about three minutes of "training" (e.g., co-occurrence counting or in-memory data structure setup) for the *RSC15-S* dataset. The factorization-based methods and the deep learning based method, on the other hand, need about 6 to 8 hours to learn a model for the single data split. Note that while the deep learning method GRU4REC and the factorization-based approach SMF

do not take the longest absolute time in this comparison, they are the only method for which the computations are done on the GPU. Running GRU4REC, for example, on a CPU tripled the computation times according to the measurements in (Jannach and Ludewig, 2017).

Looking at the times needed to compute a single recommendation list, given a session beginning, we can observe that the simple rule-based methods AR, MC, and SR are among the fastest ones with prediction times at about 3 ms for the *RSC15-S* dataset. The factorization-based methods and GRU4REC are also very efficient, with prediction times mostly below 10 ms on average. The nearest-neighbor methods are slower for this task as they have to consider the neighbors in the prediction process. Since the neighbors can be determined through fast lookup operations, the overall prediction time even for the more elaborate S-SKNN and V-SKNN similarity schemes never exceeds 33 ms for creating a recommendation list.

Looking at the *8TRACKS* dataset with its large number of items, we can, however, see that the prediction times for many algorithms, including GRU4REC and several of the factorization-based ones significantly increase, while the prediction time for the neighborhood models only doubles. In the end, making the neighborhood-based computations is at least as fast as computing the predictions based on the offline-trained models. Overall, due to the used in-memory data structures and through the neighborhood sampling approach, such neighborhood models are also suited under the narrow time constraints of real-time recommendations. Differently from other methods, newly arriving interaction signals can be easily included in the underlying model without re-training (Jugovac et al., 2018).

Memory Requirements In terms of the memory requirements, the rule-based methods AR, MC, and SR that basically record item co-occurrences of size two require the least memory, i.e., below 100 megabytes. Also the memory demands of GRU4REC are very low in this comparison, and GRU4REC occupies only about 60 MB of memory on the graphics card for the *RSC15-S* dataset. The factorization-based methods and the neighborhood methods, in contrast, have substantially higher memory requirements. The lookup data structures of the neighborhood-based methods, for example, in our implementation occupy about 6 GB of memory. When additional recency-based sampling is applied, which according to the analyses above does not hurt accuracy, these demands could, however, be substantially lowered.

For some algorithms, the memory requirements largely depend on the characteristics of the datasets. Looking at the numbers for the *8TRACKS* dataset, which covers over 300,000 different items (in contrast to the about 30,000 of the *RSC15* dataset), we see that in particular the memory demand of GRU4REC substantially increases with the number of items. As a result, GRU4REC's network even needs more memory than neighborhood-based methods for this dataset. Given these observations it seems promising to implement additional data sampling strategies within the more complex methods—as we did for the nearest neighbor methods—to decrease their computational demands.

6 Conclusion and Future Directions

6.1 Summary of Main Insights

Being able to predict the user's short-term interest in an online session is a highly relevant problem in practice, which has raised increased interest also in the academic field in recent years. Even though a number of different algorithmic approaches were proposed over the years, no standard benchmark datasets and baseline algorithms exist today. In this work, we have compared a number of very recent and computationally complex algorithms for session-based recommendation with more light-weight approaches based, e.g., on session neighborhoods. The experimental analyses on a number of different datasets show that in many cases one of the simpler methods is able to outperform even the most recent methods based on recurrent neural networks in terms of the prediction accuracy. At the same time, the computational demands of these methods can be kept comparably low when using in-memory cache data structures and data sampling.

Overall, the results, therefore, indicate that additional research is required with respect to the development of sophisticated models that are more flexible in terms of how much sequential information is contained in the training data. This is in particular the case as several improvements for the nearest-neighbor methods can be imagined as well. In this work, we could for example observe that already using a different similarity measure, as done in the V-SKNN method, can lead to substantial performance improvements for different datasets. As a side result, we noticed that using the latent feature vectors of the items of the current session for sequential factorization-based methods does not lead to high accuracy values and that such methods are usually not strong baselines when comparing session-based algorithms.

Currently, constantly new deep learning-based algorithms for session-based recommendation are proposed, e.g., (Liu et al., 2018) and (Li et al., 2018), which, for example, report improvements over GRU4REC. We performed an initial evaluation of the STAMP method proposed in (Liu et al., 2018). Our first results indicate that STAMP does not outperform the trivial SR technique in terms of the MRR on the *Diginetica* dataset that was used for the evaluation in (Liu et al., 2018). The STAMP method, however, seems to be advantageous in terms of the hit rate for this particular dataset.

Generally, it is of course surprising that a recent and popular RNN-based method is not substantially better than longer-existing nearest neighbor approaches. We believe that this might be a result of the fact that for the specific task of session-based recommendation no "standard" existed so far with respect to baseline techniques and evaluation protocols. With this work, our aim is to contribute better baselines to benchmark session-based algorithms in the future. A limitation of our work in some sense is that we could not identify *one* best baseline method across all settings and datasets. While we would identify at least one very-well performing simpler method for each dataset, the relative

performance of the algorithms seems depend on a number of factors, which are not yet fully understood.

Nevertheless, as the simple baseline approaches AR, SR, and S-KNN are computationally cheap and easy to test, their results obtained for a given dataset can potentially be used as an indicator for the general characteristics that a more complex model should aim to implement. If it, for example, turns out that SR is the best performing baseline method, GRU4REC, an extension to GRU4REC or a different sequential model might probably be a good choice. In contrast, a good performance of S-KNN indicates that a more sophisticated model should not necessarily focus too much on the order of the items in a session.

6.2 Future Directions

From an algorithmic perspective, we believe that future complex models should consider more than the last event in a session when making the next-item prediction. Even in GRU4REC, the previous items of a session are only considered implicitly through the hidden states in the prediction process. Our neighborhood models are in most cases much better when they consider all events in a session, albeit with a focus on the most recent interactions. In that context and in particular for longer sessions, it might also be helpful to detect interest changes that happen *within* an individual session. This could, for example be achieved by considering *semantic* information (e.g., meta-data or content features) about the items of the session, as was done, for example, in (Hariri et al., 2012) or (Hidasi et al., 2016b). Recent advances in the area of deep learning might be particularly helpful in this context to extract such content features, e.g., from text, images, or videos, and to use this information in *hybrid* approaches. Furthermore, the work in this paper focused on item-view events and more research is required to understand how to leverage other types of user actions like "add-to-wishlist" or "add-to-cart" in the learning and prediction process. With that information at hand, also other types of prediction problems can be addressed, e.g., whether or not a session will lead to a purchase or if there is a high probability that the user will abandon the session.

Going beyond the current session, more research also seems required in the area of *session-aware* recommendation and the consideration of previous sessions of the current user. Open questions in this area are, for example, how to model general long-term user preferences (e.g., towards certain brands in e-commerce), how to detect user-individual preference drifts, or how to identify a subset of past sessions that are good predictors for the current one. This latter aspect was for example explored in (Lerche et al., 2016) in the context of using recommendations as reminders. In addition, more elaborate strategies than static weighting schemes can be envisioned when combining short-term and long-term models. The importance weights, could for example be determined based on the length of the current session or the specific items that were considered.

Besides the consideration of signals at the individual user level, future research might also explore the incorporation of additional contextual factors or short-term trends in the community as a whole, when predicting the relevance of individual items. Recent works (Tan et al., 2016; Jannach and Ludewig, 2017; Jannach et al., 2017b) for example showed that considering short-term popularity trends and recency effects can lead to significant performance improvements in the e-commerce domain. Item recency (freshness) also plays a particular role in other domains such as music and news recommendation, and more work is required to understand how to integrate these aspects in today's recommendation algorithms.

Finally, since the relative performance of the different algorithms tested in our work sometimes varies across different datasets, more research is required to understand in which situations certain algorithms are better suited than others. These insights can then be further used to inform the design of hybrid recommendation approaches, which have shown to lead to the highest recommendation accuracy for session-based recommendation also in (Jannach and Ludewig, 2017). Generally, many factors can influence the performance of a certain recommendation algorithm. Adomavicius and Zhang (2012) have, for example, made a number of important analyses aiming to relate dataset characteristics, e.g., rating distributions and dataset sizes, with prediction accuracy. In the context of session-based recommendation problems, additional factors may have an influence, for example, the existence and strength of the sequential patterns that can be found in the data. Furthermore, often domain-specific aspects like item freshness and general item popularity might play important roles and should be further explored in future research.

References

Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Trans. on Knowl. and Data Eng.*, 24(5):896–911, May 2012.

Gediminas Adomavicius and Jingjing Zhang. Impact of data characteristics on recommender systems performance. *ACM Trans. Manage. Inf. Syst.*, 3 (1):3:1–3:17, 2012.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93*, pages 207–216, 1993.

Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. Predicting the next app that you are going to use. In *WSDM '15*, pages 285–294, 2015.

Daniel Billsus, Michael J. Pazzani, and James Chen. A learning agent for wireless news access. In *IUI '00*, pages 33–36, 2000.

Geoffray Bonnin and Dietmar Jannach. Automated generation of music playlists: Survey and experiments. *Computing Surveys*, 47(2):26:1–26:35, November 2014.

Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *KDD '12*, pages 714–722, 2012.

Shuo Chen, Jiexun Xu, and Thorsten Joachims. Multi-space probabilistic sequence modeling. In *KDD '13*, pages 865–873, 2013.

Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI '13*, pages 2605–2611, 2013.

Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP '14*, pages 1724–1734, October 2014.

Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: Scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 271–280, 2007.

James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *RecSys '10*, pages 293–296, 2010.

Robin Devooght and Hugues Bersini. Long and short-term recommendations with recurrent neural networks. In *UMAP '17*, pages 13–21, 2017.

Nemanja Djuric, Vladan Radosavljevic, Mihajlo Grbovic, and Narayan Bhamidipati. Hidden conditional random fields with deep user embeddings for ad targeting. In *ICDM '14*, pages 779–784, 2014.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD '16*, pages 1555–1564, 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12: 2121–2159, July 2011.

Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. Personalized ranking metric embedding for next new POI recommendation. In *IJCAI '15*, pages 2069–2075, 2015.

Florent Garcin, Christos Dimitrakakis, and Boi Faltings. Personalized news recommendation with context trees. In *RecSys '13*, pages 105–112, 2013.

Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. In *KDD '15*, pages 1809–1818, 2015.

Negar Hariri, Bamshad Mobasher, and Robin Burke. Context-aware music recommendation based on latent topic sequential patterns. In *RecSys '12*, pages 131–131, 2012.

Jing He, Xin Li, Lejian Liao, Dandan Song, and William Cheung. Inferring a personalized next point-of-interest recommendation model with latent behavior patterns. In *AAAI '16*, 2016.

Qi He, Daxin Jiang, Zhen Liao, Steven C. H. Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. Web query recommendation via sequential query prediction. In *ICDE '09*, pages 1443–1454, 2009.

Ruining He and Julian McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. *CoRR*, abs/1609.09152, 2016. URL `https://arxiv.org/abs/1609.09152`.

Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. *CoRR*, abs/1706.03847, 2017. URL `http://arxiv.org/abs/1706.03847`.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *ICLR '16*, 2016a.

Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *RecSys '16*, pages 241–248, 2016b.

Mehdi Hosseinzadeh Aghdam, Negar Hariri, Bamshad Mobasher, and Robin Burke. Adapting recommendations to contextual changes using hierarchical hidden markov models. In *RecSys '15*, pages 241–244, 2015.

Dietmar Jannach and Gediminas Adomavicius. Recommendations with a purpose. In *RecSys '16*, pages 7–10, 2016.

Dietmar Jannach and Kolja Hegelich. A case study on the effectiveness of recommendations in the mobile internet. In *RecSys '09*, pages 205–208, 2009.

Dietmar Jannach and Malte Ludewig. When recurrent neural networks meet the neighborhood for session-based recommendation. In *RecSys '17*, pages 306–310, 2017.

Dietmar Jannach, Lukas Lerche, and Michael Jugovac. Adaptation and evaluation of recommendations for short-term shopping goals. In *RecSys '15*, pages 211–218, 2015a.

Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction*, 25 (5):427–491, 2015b.

Dietmar Jannach, Iman Kamehkhosh, and Lukas Lerche. Leveraging multi-dimensional user models for personalized next-track music recommendation. In *ACM SAC 2017*, 2017a.

Dietmar Jannach, Malte Ludewig, and Lukas Lerche. Session-based item recommendation in e-commerce: On short-term intents, reminders, trends, and discounts. *User-Modeling and User-Adapted Interaction*, 27(3–5):351–392, 2017b.

Michael Jugovac, Dietmar Jannach, and Mozhgan Karimi. StreamingRec: A Framework for Benchmarking Stream-based News Recommenders. In *RecSys 2018*, 2018.

Santosh Kabbur, Xia Ning, and George Karypis. FISM: Factored item similarity models for top-n recommender systems. In *KDD '13*, pages 659–667, 2013.

Iman Kamehkhosh, Dietmar Jannach, and Malte Ludewig. A comparison of frequent pattern techniques and a deep learning method for session-based recommendation. In *TempRec Workshop at ACM RecSys '17*, Como, Italy, 2017.

Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. News recommender systems - survey and roads ahead. *Information Processing and Management*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

Dokyun Lee and Kartik Hosanagar. Impact of recommender systems on sales volume and diversity. In *ICIS 2014*, 2014.

Lukas Lerche, Dietmar Jannach, and Malte Ludewig. On the value of reminders within e-commerce recommendations. In *UMAP '16*, pages 27–35, 2016.

Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. Learning from History and Present: Next-item Recommendation via Discriminatively Exploiting User Behaviors. In *KDD 2018*, 2018.

Defu Lian, Vincent W. Zheng, and Xing Xie. Collaborative filtering meets next check-in location prediction. In *WWW '13*, pages 231–232, 2013.

Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *IUI '10*, pages 31–40, 2010.

Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *KDD 2018*, 2018.

Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. Unified point-of-interest recommendation with temporal interval assessment. In *KDD '16*, pages 1015–1024, 2016.

Cornelius A. Ludmann. Recommending news articles in the CLEF news recommendation evaluation lab with the data stream management system odysseus. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation*, 2017.

Brian Mcfee and Gert Lanckriet. The natural language of playlists. In *ISMIR '11*, pages 537–541, 2011.

Brian McFee and Gert R. G. Lanckriet. Hypergraph models of playlist dialects. In *ISMIR '12*, pages 343–348, 2012.

Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *ICDM '02*, pages 669–672, 2002.

Omar Moling, Linas Baltrunas, and Francesco Ricci. Optimal radio channel recommendations with explicit and implicit feedback. In *RecSys '12*, pages 75–82, 2012.

Nagarajan Natarajan, Donghyuk Shin, and Inderjit S. Dhillon. Which app will you use next?: Collaborative filtering with interactional context. In *RecSys '13*, pages 201–208, 2013.

J.R. Norris. *Markov Chains*. Cambridge University Press, Cambridge, 1997.

Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *RecSys '17*, 2017.

Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Computing Surveys*, 54:1–36, 2018.

Siddharth Reddy, Igor Labutov, and Thorsten Joachims. Learning student and content embeddings for personalized lesson sequence recommendation. In *ACM Learning @ Scale '16*, pages 93–96, 2016.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI '09*, pages 452–461, 2009.

Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW '10*, pages 811–820, 2010.

Guy Shani, David Heckerman, and Ronen I. Brafman. An MDP-based recommender system. *J. Mach. Learn. Res.*, 6:1265–1295, 2005.

Harold Soh, Scott Sanner, Madeleine White, and Greg Jamieson. Deep sequential recommendation for personalized adaptive user interfaces. In *IUI '17*, pages 589–593, 2017.

Qiang Song, Jian Cheng, Ting Yuan, and Hanqing Lu. Personalized recommendation meets your next favorite. In *CIKM '15*, pages 1775–1778, 2015.

Yang Song, Ali Mamdouh Elkahky, and Xiaodong He. Multi-rate deep learning for temporal recommendation. In *SIGIR '16*, pages 909–912, 2016.

Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM '15*, pages 553–562, 2015.

Yukihiro Tagami, Hayato Kobayashi, Shingo Ono, and Akira Tajima. Modeling user activities on the web using paragraph vector. In *WWW '15*, pages 125–126, 2015.

Yong Kiam Tan, Xinxing Xu, and Yong Liu. Improved Recurrent Neural Networks for Session-based Recommendations. In *DLRS '16 Workshop at ACM RecSys*, pages 17–22, 2016.

Maryam Tavakol and Ulf Brefeld. Factored mdps for detecting topics of user sessions. In *RecSys '14*, pages 33–40, 2014.

Roberto Turrin, Massimo Quadrana, Andrea Condorelli, Roberto Pagano, and Paolo Cremonesi. 30music listening and playlists dataset. In *Poster Proceedings of RecSys '15*, 2015.

Bartlomiej Twardowski. Modelling contextual information in session-aware recommender systems with neural networks. In *RecSys '16*, pages 273–276, 2016.

Flavian Vasile, Elena Smirnova, and Alexis Conneau. Meta-prod2vec: Product embeddings using side-information for recommendation. In *RecSys '16*, pages 225–232, 2016.

Koen Verstrepen and Bart Goethals. Unifying nearest neighbors collaborative filtering. In *RecSys '14*, pages 177–184, 2014.

Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, and Guoping Hu. Personalized next-song recommendation in online karaokes. In *RecSys '13*, pages 137–140, 2013.

Ghim-Eng Yap, Xiao-Li Li, and Philip S. Yu. Effective next-items recommendation via personalized sequential pattern mining. In *DASFAA '12, Volume Part II*, pages 48–64, 2012.

Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. A dynamic recurrent model for next basket recommendation. In *SIGIR '16*, pages 729–732, 2016.

Eva Zangerle, Martin Pichl, Wolfgang Gassler, and Günther Specht. #Nowplaying Music Dataset: Extracting Listening Behavior from Twitter. In *WISMM '14 Workshop at MM '14*, pages 21–26, 2014.

Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL `http://arxiv.org/abs/1212.5701`.

Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI '14*, pages 1369–1375, 2014.

Elena Zheleva, John Guiver, Eduarda Mendes Rodrigues, and Nataša Milić-Frayling. Statistical models of music-listening sessions in social media. In *WWW '10*, pages 1019–1028, 2010.

Author Biographies

Malte Ludewig is a PhD candidate in Computer Science at TU Dortmund, Germany, from where he also received his MSc degree. His research interests lie in the field of recommender systems—with a focus on session-based recommendations—and personalization in e-commerce environments in general.

Dietmar Jannach is a Professor of Computer Science at AAU Klagenfurt, Austria, and head of the department's information systems research group. Dr. Jannach has worked on different areas of artificial intelligence, including recommender systems, model-based diagnosis, and knowledge-based systems. He is the leading author of a textbook on recommender systems and has authored more than hundred research papers, focusing on the application of artificial intelligence technology to practical problems.

A Parameter Configurations

Table 9: Parameters for algorithm GRU4REC for all datasets.

Dataset	layer_size	objective	lr	momentum	drop_out
RSC15	100	BPR_{max}	0.20	0.5	0.0
TMALL	100	$TOP1_{max}$	0.05	0.0	0.3
RETAILROCKET	100	$TOP1_{max}$	0.15	0.3	0.0
8TRACKS	100	$TOP1_{max}$	0.10	0.0	0.7
AOTM	100	BPR_{max}	0.10	0.3	0.1
NOWPLAYING	100	BPR_{max}	0.10	0.5	0.1
30MUSIC	100	$TOP1_{max}$	0.10	0.1	0.1
ZALANDO	100	BPR_{max}	0.20	0.1	0.1
CLEF	100	$TOP1_{max}$	0.20	0.2	0.5
LASTFM	100	BPR_{max}	0.15	0.4	0.3

Table 10: Parameters used for the SMF algorithm for all datasets.

Dataset	layer_size	objective	lr	momentum	drop_out	skip
RSC15	100	$TOP1_{max}$	0.085	0.2	0.30	0.00
TMALL	100	BPR_{max}	0.015	0.6	0.00	0.00
RETAILROCKET	100	BPR_{max}	0.045	0.1	0.40	0.20
8TRACKS	100	$TOP1_{max}$	0.010	0.5	0.30	0.35
AOTM	100	BPR_{max}	0.090	0.8	0.40	0.20
NOWPLAYING	100	$TOP1_{max}$	0.055	0.2	0.40	0.20
30MUSIC	100	$TOP1_{max}$	0.100	0.1	0.20	0.20
ZALANDO	100	$TOP1_{max}$	0.030	0.3	0.25	0.00
CLEF	100	BPR_{max}	0.050	0.0	0.40	0.25
LASTFM	100	$TOP1_{max}$	0.015	0.3	0.15	0.45

Table 11: Parameters used for the V-SKNN algorithm for all datasets.

Dataset	K	samples
RSC15	200	2000
TMALL	200	2000
RETAILROCKET	200	2000
ZALANDO	200	2000
8TRACKS	200	2000
AOTM	200	2000
NOWPLAYING	100	1000
30MUSIC	100	1000
CLEF	100	1000
LASTFM	200	2000

Table 12: Parameters used for the SKNN, S-SKNN, and SF-SKNN algorithm for all datasets.

Dataset	K	samples
RSC15	500	1000
TMALL	100	500
RETAILROCKET	100	500
ZALANDO	100	500
8TRACKS	100	500
AOTM	100	500
NOWPLAYING	100	500
30MUSIC	100	500
CLEF	100	500
LASTFM	100	500

B Full Result Tables

Table 13: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the RSC15 dataset (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
GRU4REC	0.308	0.683	0.504	0.054	0.301	0.591	0.431	0.058
SR	0.304	0.653	0.668	0.072	0.298	0.569	0.592	0.073
SMF	0.302	0.666	0.565	0.055	0.295	0.575	0.486	0.058
MC	0.300	0.642	0.645	0.070	0.295	0.562	0.584	0.071
AR	0.289	0.636	0.630	0.093	0.283	0.550	0.548	0.091
V-SKNN	0.283	0.653	0.619	0.079	0.277	0.563	0.534	0.081
S-SKNN	0.272	0.602	0.655	0.072	0.267	0.531	0.543	0.077
SF-SKNN	0.270	0.589	0.619	0.066	0.266	0.524	0.545	0.074
S-KNN	0.266	0.621	0.634	0.073	0.259	0.526	0.520	0.078
IKNN	0.208	0.486	0.755	0.041	0.203	0.408	0.671	0.046
FPMC	0.201	0.363	0.975	0.055	0.198	0.311	0.908	0.056
BPR-MF	0.176	0.235	0.911	0.088	0.175	0.223	0.793	0.079
FISM	0.115	0.162	0.974	0.008	0.114	0.149	0.917	0.012
FOSSIL	0.062	0.190	0.917	0.048	0.058	0.135	0.806	0.047
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
GRU4REC	0.285	0.470	0.355	0.062	0.263	0.371	0.180	0.180
SR	0.283	0.457	0.503	0.075	0.262	0.365	0.182	0.182
SMF	0.280	0.459	0.406	0.061	0.258	0.364	0.177	0.177
MC	0.280	0.452	0.506	0.073	0.259	0.362	0.181	0.181
AR	0.268	0.438	0.455	0.092	0.248	0.348	0.171	0.171
V-SKNN	0.261	0.446	0.451	0.084	0.239	0.351	0.154	0.154
S-SKNN	0.252	0.424	0.437	0.082	0.231	0.333	0.153	0.153
SF-SKNN	0.252	0.421	0.457	0.081	0.232	0.332	0.154	0.154
S-KNN	0.244	0.411	0.417	0.084	0.224	0.322	0.149	0.149
IKNN	0.190	0.315	0.566	0.050	0.174	0.244	0.121	0.121
FPMC	0.191	0.261	0.774	0.058	0.183	0.226	0.148	0.148
BPR-MF	0.174	0.214	0.630	0.070	0.172	0.205	0.144	0.144
FISM	0.112	0.135	0.810	0.019	0.110	0.126	0.096	0.096
FOSSIL	0.052	0.092	0.653	0.046	0.047	0.067	0.031	0.031

Table 14: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the RSC15-S dataset (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
SMF	0.309	0.713	0.512	0.052	0.301	0.606	0.414	0.054
GRU4REC	0.308	0.719	0.350	0.033	0.301	0.609	0.277	0.034
SR	0.308	0.690	0.512	0.038	0.301	0.591	0.407	0.038
MC	0.296	0.667	0.518	0.039	0.289	0.567	0.413	0.039
AR	0.281	0.655	0.473	0.045	0.273	0.543	0.374	0.043
V-SKNN	0.274	0.675	0.427	0.037	0.266	0.562	0.328	0.039
S-SKNN	0.266	0.667	0.417	0.035	0.258	0.548	0.309	0.038
SF-SKNN	0.260	0.670	0.446	0.037	0.251	0.545	0.339	0.039
S-KNN	0.250	0.641	0.398	0.036	0.242	0.521	0.293	0.038
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
SMF	0.284	0.476	0.320	0.056	0.259	0.368	0.177	0.177
GRU4REC	0.283	0.473	0.207	0.037	0.259	0.369	0.175	0.175
SR	0.284	0.468	0.308	0.040	0.262	0.371	0.179	0.179
MC	0.273	0.444	0.314	0.041	0.252	0.354	0.174	0.174
AR	0.257	0.422	0.283	0.046	0.236	0.333	0.163	0.163
V-SKNN	0.248	0.429	0.242	0.042	0.225	0.329	0.145	0.145
S-SKNN	0.240	0.414	0.221	0.041	0.218	0.318	0.142	0.142
SF-SKNN	0.233	0.406	0.246	0.042	0.210	0.307	0.137	0.137
S-KNN	0.224	0.390	0.207	0.041	0.203	0.297	0.133	0.133

Table 15: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the TMALL dataset (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
S-SKNN	0.185	0.387	0.467	0.025	0.181	0.330	0.309	0.027
S-KNN	0.182	0.404	0.381	0.026	0.177	0.334	0.249	0.029
V-SKNN	0.179	0.373	0.464	0.024	0.175	0.312	0.320	0.026
BPR-MF	0.159	0.204	0.723	0.057	0.159	0.197	0.534	0.076
SF-SKNN	0.136	0.216	0.436	0.018	0.135	0.203	0.338	0.022
GRU4REC	0.129	0.277	0.151	0.035	0.125	0.225	0.109	0.039
AR	0.129	0.262	0.509	0.021	0.126	0.217	0.358	0.024
SR	0.128	0.242	0.569	0.021	0.125	0.206	0.421	0.023
SMF	0.121	0.261	0.261	0.036	0.118	0.213	0.193	0.039
MC	0.116	0.200	0.498	0.019	0.114	0.178	0.391	0.022
FPMC	0.101	0.119	0.880	0.005	0.100	0.114	0.730	0.007
IKNN	0.051	0.150	0.728	0.007	0.048	0.112	0.575	0.008
FISM	0.024	0.037	0.752	0.003	0.023	0.032	0.586	0.003
FOSSIL	0.001	0.004	0.598	0.016	0.001	0.003	0.457	0.021
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
S-SKNN	0.173	0.267	0.196	0.031	0.161	0.217	0.119	0.119
S-KNN	0.168	0.264	0.161	0.032	0.156	0.212	0.113	0.113
V-SKNN	0.167	0.251	0.218	0.029	0.157	0.207	0.118	0.118
BPR-MF	0.157	0.189	0.343	0.097	0.156	0.181	0.134	0.134
SF-SKNN	0.132	0.182	0.243	0.026	0.127	0.160	0.101	0.101
GRU4REC	0.119	0.177	0.078	0.043	0.112	0.145	0.086	0.086
AR	0.120	0.175	0.235	0.027	0.113	0.145	0.089	0.089
SR	0.120	0.170	0.286	0.026	0.114	0.144	0.091	0.091
SMF	0.112	0.168	0.140	0.041	0.105	0.138	0.079	0.079
MC	0.111	0.151	0.284	0.025	0.106	0.131	0.086	0.086
FPMC	0.100	0.109	0.540	0.010	0.099	0.105	0.093	0.093
IKNN	0.044	0.079	0.403	0.009	0.039	0.058	0.025	0.025
FISM	0.023	0.028	0.419	0.004	0.022	0.026	0.019	0.019
FOSSIL	0.001	0.002	0.310	0.028	0.001	0.002	0.001	0.001

Table 16: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the RETAILROCKET dataset (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
S-SKNN	0.345	0.591	0.596	0.056	0.341	0.537	0.480	0.066
V-SKNN	0.338	0.573	0.575	0.060	0.334	0.519	0.474	0.069
S-KNN	0.337	0.583	0.566	0.058	0.333	0.528	0.445	0.068
BPR-MF	0.303	0.357	0.824	0.060	0.303	0.352	0.627	0.072
FPMC	0.273	0.320	0.929	0.022	0.272	0.309	0.777	0.026
SF-SKNN	0.260	0.358	0.403	0.035	0.259	0.350	0.373	0.046
SR	0.245	0.419	0.524	0.042	0.243	0.386	0.458	0.050
GRU4REC	0.243	0.480	0.602	0.060	0.238	0.415	0.478	0.066
AR	0.241	0.439	0.544	0.053	0.238	0.390	0.449	0.061
MC	0.230	0.359	0.411	0.035	0.228	0.343	0.383	0.045
SMF	0.225	0.459	0.449	0.085	0.221	0.393	0.360	0.092
IKNN	0.107	0.240	0.584	0.033	0.105	0.202	0.505	0.038
FISM	0.075	0.132	0.848	0.018	0.074	0.112	0.672	0.019
FOSSIL	0.022	0.058	0.753	0.127	0.020	0.043	0.560	0.150
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
S-SKNN	0.332	0.470	0.344	0.076	0.318	0.406	0.247	0.247
V-SKNN	0.326	0.455	0.348	0.078	0.312	0.396	0.245	0.245
S-KNN	0.324	0.458	0.316	0.080	0.310	0.396	0.242	0.242
BPR-MF	0.302	0.345	0.417	0.083	0.300	0.337	0.267	0.267
FPMC	0.271	0.298	0.560	0.032	0.269	0.289	0.251	0.251
SF-SKNN	0.257	0.331	0.311	0.058	0.250	0.302	0.208	0.208
SR	0.236	0.337	0.354	0.059	0.225	0.288	0.176	0.176
GRU4REC	0.229	0.345	0.350	0.072	0.215	0.285	0.161	0.161
AR	0.230	0.331	0.333	0.071	0.218	0.280	0.170	0.170
MC	0.224	0.308	0.322	0.056	0.215	0.270	0.171	0.171
SMF	0.211	0.322	0.270	0.099	0.198	0.264	0.148	0.148
IKNN	0.099	0.159	0.388	0.042	0.091	0.127	0.065	0.065
FISM	0.071	0.094	0.474	0.023	0.069	0.083	0.058	0.058
FOSSIL	0.019	0.032	0.377	0.171	0.017	0.024	0.012	0.012

Table 17: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the ZALANDO dataset (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
SR	0.304	0.483	0.586	0.061	0.302	0.462	0.433	0.066
MC	0.303	0.455	0.513	0.060	0.302	0.441	0.412	0.066
IKNN	0.275	0.405	0.714	0.037	0.273	0.385	0.532	0.041
GRU4REC	0.267	0.468	0.304	0.101	0.265	0.433	0.239	0.103
SMF	0.267	0.447	0.362	0.107	0.265	0.418	0.282	0.108
AR	0.258	0.467	0.467	0.089	0.256	0.435	0.337	0.090
SF-SKNN	0.249	0.438	0.432	0.057	0.249	0.430	0.348	0.068
V-SKNN	0.233	0.521	0.432	0.096	0.230	0.482	0.296	0.096
S-SKNN	0.219	0.499	0.435	0.087	0.216	0.456	0.280	0.092
S-KNN	0.172	0.456	0.309	0.093	0.167	0.380	0.201	0.097
BPR-MF	0.104	0.162	0.609	0.058	0.103	0.152	0.415	0.069
FPMC	0.051	0.075	0.812	0.021	0.050	0.067	0.629	0.022
FISM	0.004	0.011	0.624	0.020	0.004	0.008	0.444	0.020
FOSSIL	0.002	0.005	0.671	0.034	0.002	0.004	0.493	0.036
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
SR	0.298	0.429	0.290	0.069	0.287	0.382	0.211	0.211
MC	0.298	0.415	0.292	0.069	0.289	0.377	0.218	0.218
IKNN	0.270	0.362	0.349	0.047	0.264	0.335	0.205	0.205
GRU4REC	0.259	0.389	0.182	0.100	0.247	0.337	0.177	0.177
SMF	0.259	0.380	0.210	0.104	0.249	0.333	0.183	0.183
AR	0.250	0.393	0.233	0.088	0.237	0.338	0.159	0.159
SF-SKNN	0.245	0.403	0.249	0.074	0.232	0.348	0.142	0.142
V-SKNN	0.222	0.422	0.197	0.092	0.205	0.346	0.095	0.095
S-SKNN	0.207	0.388	0.174	0.095	0.189	0.311	0.095	0.095
S-KNN	0.154	0.290	0.125	0.103	0.137	0.216	0.079	0.079
BPR-MF	0.102	0.141	0.247	0.083	0.099	0.130	0.073	0.073
FPMC	0.049	0.061	0.434	0.025	0.048	0.056	0.042	0.042
FISM	0.004	0.006	0.290	0.021	0.004	0.005	0.003	0.003
FOSSIL	0.002	0.003	0.333	0.037	0.002	0.002	0.001	0.001

Table 18: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the 8TRACKS dataset (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
AR	0.0071	0.0255	0.4530	0.0912	0.0066	0.0173	0.3178	0.1075
SMF	0.0064	0.0231	0.1528	0.0864	0.0058	0.0148	0.1076	0.0916
SR	0.0064	0.0171	0.4967	0.0531	0.0061	0.0125	0.3645	0.0636
SF-SKNN	0.0064	0.0119	0.3049	0.0363	0.0063	0.0102	0.2439	0.0517
V-SKNN	0.0057	0.0352	0.4081	0.1194	0.0048	0.0211	0.2523	0.1353
S-KNN	0.0053	0.0376	0.2431	0.1080	0.0041	0.0198	0.1544	0.1153
IKNN	0.0051	0.0177	0.6956	0.0245	0.0047	0.0124	0.5101	0.0267
GRU4REC	0.0050	0.0189	0.0693	0.1223	0.0045	0.0118	0.0512	0.1326
S-SKNN	0.0048	0.0293	0.4509	0.0807	0.0040	0.0182	0.2741	0.0961
MC	0.0046	0.0099	0.3496	0.0321	0.0045	0.0079	0.2756	0.0402
BPR-MF	0.0002	0.0004	0.6138	0.0088	0.0002	0.0003	0.4253	0.0131
FOSSIL	0.0001	0.0002	0.6703	0.0084	0.0001	0.0002	0.4941	0.0121
FPMC	0.0000	0.0001	0.7608	0.0031	0.0000	0.0001	0.5729	0.0035
FISM	0.0000	0.0001	0.6210	0.0027	0.0000	0.0000	0.4460	0.0028
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
AR	0.0057	0.0108	0.1998	0.1251	0.0049	0.0073	0.0032	0.0032
SMF	0.0051	0.0092	0.0753	0.0958	0.0045	0.0064	0.0031	0.0031
SR	0.0056	0.0090	0.2395	0.0744	0.0051	0.0069	0.0038	0.0038
SF-SKNN	0.0060	0.0083	0.1794	0.0690	0.0057	0.0071	0.0047	0.0047
V-SKNN	0.0034	0.0102	0.1496	0.1364	0.0022	0.0048	0.0004	0.0004
S-KNN	0.0026	0.0079	0.0975	0.1065	0.0016	0.0033	0.0004	0.0004
IKNN	0.0041	0.0078	0.3293	0.0286	0.0035	0.0053	0.0022	0.0022
GRU4REC	0.0039	0.0070	0.0385	0.1414	0.0034	0.0050	0.0023	0.0023
S-SKNN	0.0026	0.0081	0.1404	0.1028	0.0016	0.0034	0.0004	0.0004
MC	0.0043	0.0062	0.2022	0.0493	0.0040	0.0051	0.0032	0.0032
BPR-MF	0.0002	0.0003	0.2576	0.0203	0.0002	0.0002	0.0001	0.0001
FOSSIL	0.0001	0.0001	0.3338	0.0178	0.0001	0.0001	0.0001	0.0001
FPMC	0.0000	0.0001	0.3879	0.0043	0.0000	0.0000	0.0000	0.0000
FISM	0.0000	0.0000	0.2952	0.0030	0.0000	0.0000	0.0000	0.0000

Table 19: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the AOTM dataset (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
SMF	0.0111	0.0298	0.2457	0.1998	0.0105	0.0205	0.1795	0.2085
SF-SKNN	0.0111	0.0145	0.3559	0.0508	0.0110	0.0139	0.3022	0.0686
SR	0.0077	0.0195	0.5864	0.0533	0.0073	0.0149	0.4481	0.0599
GRU4REC	0.0072	0.0157	0.4653	0.1151	0.0070	0.0125	0.3550	0.1131
MC	0.0063	0.0133	0.3803	0.0498	0.0062	0.0112	0.3198	0.0602
AR	0.0059	0.0233	0.5532	0.1049	0.0053	0.0146	0.4003	0.1178
V-SKNN	0.0055	0.0378	0.5363	0.1397	0.0043	0.0208	0.3357	0.1550
S-SKNN	0.0055	0.0397	0.5357	0.1289	0.0042	0.0209	0.3228	0.1475
S-KNN	0.0054	0.0429	0.2802	0.1678	0.0038	0.0200	0.1785	0.1666
IKNN	0.0049	0.0187	0.7880	0.0473	0.0045	0.0122	0.5777	0.0481
FOSSIL	0.0007	0.0027	0.5529	0.0978	0.0006	0.0017	0.3717	0.1139
BPR-MF	0.0005	0.0018	0.5659	0.0968	0.0005	0.0012	0.3550	0.1180
FPMC	0.0003	0.0007	0.7851	0.0264	0.0003	0.0006	0.5867	0.0289
FISM	0.0001	0.0004	0.6172	0.0272	0.0001	0.0002	0.4296	0.0288

Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
SMF	0.0097	0.0149	0.1265	0.2136	0.0091	0.0118	0.0070	0.0070
SF-SKNN	0.0109	0.0130	0.2306	0.0875	0.0107	0.0121	0.0096	0.0096
SR	0.0068	0.0107	0.3015	0.0626	0.0062	0.0081	0.0047	0.0047
GRU4REC	0.0067	0.0102	0.2432	0.1141	0.0063	0.0085	0.0045	0.0045
MC	0.0059	0.0089	0.2449	0.0681	0.0055	0.0072	0.0042	0.0042
AR	0.0046	0.0089	0.2543	0.1318	0.0039	0.0059	0.0024	0.0024
V-SKNN	0.0027	0.0085	0.1927	0.1592	0.0016	0.0034	0.0004	0.0004
S-SKNN	0.0025	0.0077	0.1718	0.1558	0.0014	0.0028	0.0005	0.0005
S-KNN	0.0021	0.0063	0.1108	0.1549	0.0011	0.0022	0.0005	0.0005
IKNN	0.0038	0.0073	0.3591	0.0500	0.0033	0.0051	0.0020	0.0020
FOSSIL	0.0005	0.0010	0.2311	0.1308	0.0005	0.0007	0.0003	0.0003
BPR-MF	0.0004	0.0007	0.1900	0.1403	0.0004	0.0005	0.0003	0.0003
FPMC	0.0003	0.0004	0.3884	0.0325	0.0003	0.0003	0.0003	0.0003
FISM	0.0001	0.0001	0.2743	0.0311	0.0000	0.0000	0.0000	0.0000

Table 20: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the 30MUSIC dataset (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
SR	0.2377	0.3323	0.3893	0.0232	0.2363	0.3120	0.2913	0.0273
MC	0.2318	0.2844	0.2038	0.0205	0.2314	0.2780	0.1804	0.0265
GRU4REC	0.2264	0.3257	0.3447	0.0556	0.2249	0.3042	0.2423	0.0567
SF-SKNN	0.2079	0.2856	0.1854	0.0219	0.2078	0.2834	0.1634	0.0302
SMF	0.1777	0.2843	0.1508	0.1048	0.1756	0.2547	0.1117	0.1062
V-SKNN	0.1099	0.3819	0.3170	0.0538	0.1040	0.3002	0.1944	0.0573
IKNN	0.1086	0.2971	0.4596	0.0226	0.1053	0.2501	0.3122	0.0249
S-SKNN	0.1077	0.3856	0.2931	0.0515	0.1014	0.2975	0.1759	0.0569
AR	0.0960	0.3088	0.3524	0.0394	0.0911	0.2395	0.2375	0.0433
S-KNN	0.0898	0.3443	0.1912	0.0574	0.0832	0.2501	0.1155	0.0637
BPR-MF	0.0427	0.0580	0.4521	0.0281	0.0425	0.0548	0.2792	0.0381
FPMC	0.0293	0.0359	0.6544	0.0078	0.0291	0.0334	0.4556	0.0085
FISM	0.0029	0.0047	0.4676	0.0084	0.0028	0.0038	0.3052	0.0089
FOSSIL	0.0029	0.0100	0.3347	0.0297	0.0027	0.0073	0.1919	0.0436

Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
SR	0.2326	0.2845	0.1920	0.0294	0.2270	0.2601	0.2005	0.2005
MC	0.2298	0.2663	0.1467	0.0309	0.2270	0.2544	0.2043	0.2043
GRU4REC	0.2215	0.2796	0.1629	0.0557	0.2162	0.2564	0.1835	0.1835
SF-SKNN	0.2062	0.2726	0.1318	0.0371	0.2009	0.2499	0.1614	0.1614
SMF	0.1712	0.2223	0.0817	0.1057	0.1655	0.1970	0.1405	0.1405
V-SKNN	0.0882	0.1813	0.1165	0.0612	0.0727	0.1125	0.0442	0.0442
IKNN	0.0956	0.1788	0.1961	0.0248	0.0837	0.1265	0.0523	0.0523
S-SKNN	0.0851	0.1753	0.1043	0.0629	0.0701	0.1086	0.0428	0.0428
AR	0.0803	0.1580	0.1466	0.0476	0.0686	0.1063	0.0413	0.0413
S-KNN	0.0689	0.1424	0.0691	0.0714	0.0566	0.0877	0.0344	0.0344
BPR-MF	0.0421	0.0515	0.1523	0.0518	0.0414	0.0483	0.0356	0.0356
FPMC	0.0289	0.0317	0.2872	0.0096	0.0286	0.0303	0.0272	0.0272
FISM	0.0028	0.0034	0.1859	0.0095	0.0027	0.0030	0.0025	0.0025
FOSSIL	0.0023	0.0048	0.0914	0.0634	0.0019	0.0031	0.0011	0.0011

Table 21: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the NOWPLAYING dataset (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
SR	0.1053	0.2033	0.4656	0.0247	0.1031	0.1712	0.3605	0.0284
GRU4REC	0.1018	0.1970	0.4331	0.0516	0.0995	0.1632	0.3261	0.0529
MC	0.0971	0.1582	0.2936	0.0284	0.0960	0.1417	0.2547	0.0347
SF-SKNN	0.0954	0.1647	0.2773	0.0311	0.0945	0.1524	0.2369	0.0414
SMF	0.0882	0.1825	0.2417	0.0916	0.0859	0.1484	0.1847	0.0960
V-SKNN	0.0785	0.2552	0.4283	0.0639	0.0737	0.1861	0.2904	0.0719
S-SKNN	0.0777	0.2622	0.4149	0.0624	0.0725	0.1880	0.2727	0.0699
AR	0.0710	0.2076	0.4531	0.0511	0.0672	0.1518	0.3261	0.0584
S-KNN	0.0689	0.2429	0.3007	0.0690	0.0637	0.1676	0.1962	0.0759
IKNN	0.0569	0.1822	0.5799	0.0294	0.0534	0.1321	0.4313	0.0308
BPR-MF	0.0392	0.0621	0.5903	0.0672	0.0387	0.0547	0.3764	0.0843
FPMC	0.0331	0.0470	0.7865	0.0154	0.0327	0.0418	0.5833	0.0190
FOSSIL	0.0136	0.0432	0.5950	0.0336	0.0127	0.0302	0.4035	0.0389
FISM	0.0108	0.0183	0.6451	0.0110	0.0105	0.0145	0.4551	0.0123
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
SR	0.0988	0.1395	0.2490	0.0305	0.0938	0.1173	0.0760	0.0760
GRU4REC	0.0958	0.1352	0.2282	0.0540	0.0913	0.1154	0.0726	0.0726
MC	0.0935	0.1236	0.2024	0.0409	0.0904	0.1099	0.0751	0.0751
SF-SKNN	0.0921	0.1344	0.1856	0.0518	0.0876	0.1149	0.0665	0.0665
SMF	0.0818	0.1181	0.1358	0.0985	0.0774	0.0985	0.0614	0.0614
V-SKNN	0.0651	0.1213	0.1819	0.0786	0.0562	0.0819	0.0381	0.0381
S-SKNN	0.0632	0.1181	0.1657	0.0771	0.0544	0.0790	0.0371	0.0371
AR	0.0611	0.1060	0.2114	0.0672	0.0543	0.0763	0.0379	0.0379
S-KNN	0.0556	0.1048	0.1226	0.0825	0.0476	0.0695	0.0318	0.0318
IKNN	0.0477	0.0884	0.2869	0.0317	0.0416	0.0615	0.0265	0.0265
BPR-MF	0.0378	0.0482	0.2043	0.0995	0.0367	0.0433	0.0314	0.0314
FPMC	0.0321	0.0371	0.3807	0.0238	0.0316	0.0346	0.0293	0.0293
FOSSIL	0.0113	0.0195	0.2502	0.0439	0.0102	0.0148	0.0069	0.0069
FISM	0.0102	0.0122	0.2934	0.0143	0.0100	0.0111	0.0092	0.0092

Table 22: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the CLEF dataset (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
SMF	0.234	0.706	0.650	0.083	0.222	0.529	0.582	0.097
MC	0.225	0.687	0.732	0.095	0.213	0.514	0.705	0.123
V-SKNN	0.224	0.776	0.621	0.082	0.211	0.596	0.566	0.113
SR	0.223	0.672	0.655	0.093	0.212	0.513	0.608	0.123
GRU4REC	0.220	0.568	0.174	0.094	0.212	0.462	0.129	0.118
S-KNN	0.219	0.778	0.613	0.084	0.205	0.588	0.545	0.122
AR	0.216	0.666	0.724	0.100	0.204	0.490	0.656	0.148
IKNN	0.188	0.596	0.746	0.059	0.177	0.436	0.722	0.047
FPMC	0.171	0.598	0.847	0.082	0.159	0.414	0.721	0.093
FOSSIL	0.166	0.571	0.963	0.079	0.155	0.417	0.864	0.093
FISM	0.129	0.403	0.997	0.080	0.122	0.297	0.963	0.108
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
SMF	0.198	0.354	0.511	0.109	0.176	0.255	0.117	0.117
MC	0.190	0.339	0.653	0.144	0.170	0.249	0.113	0.113
V-SKNN	0.185	0.404	0.495	0.154	0.154	0.264	0.076	0.076
SR	0.189	0.337	0.542	0.146	0.169	0.246	0.111	0.111
GRU4REC	0.195	0.331	0.101	0.138	0.177	0.252	0.118	0.118
S-KNN	0.179	0.394	0.476	0.164	0.148	0.254	0.070	0.070
AR	0.183	0.339	0.559	0.215	0.161	0.242	0.102	0.102
IKNN	0.159	0.300	0.669	0.046	0.138	0.209	0.084	0.084
FPMC	0.138	0.258	0.601	0.108	0.120	0.178	0.078	0.078
FOSSIL	0.136	0.268	0.703	0.101	0.119	0.195	0.064	0.064
FISM	0.109	0.201	0.857	0.140	0.096	0.142	0.064	0.064

C Additional Results for Precision and Recall

Table 23: Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the TMALL dataset (sorted by P@20).

Algorithm	P@20	R@20	P@10	R@10	P@5	R@5	P@3	R@3
S-KNN	0.095	0.312	0.141	0.257	0.196	0.199	0.235	0.156
S-SKNN	0.094	0.263	0.139	0.215	0.191	0.165	0.229	0.129
V-SKNN	0.091	0.291	0.131	0.239	0.186	0.187	0.230	0.150
SMF	0.068	0.230	0.099	0.184	0.139	0.141	0.172	0.113
GRU4REC	0.068	0.233	0.098	0.187	0.137	0.143	0.170	0.115
AR	0.057	0.173	0.082	0.138	0.115	0.106	0.143	0.085
SR	0.052	0.193	0.081	0.162	0.121	0.131	0.158	0.109
IKNN	0.043	0.112	0.059	0.082	0.077	0.057	0.091	0.042
SF-SKNN	0.041	0.136	0.072	0.125	0.116	0.108	0.154	0.092
MC	0.036	0.124	0.058	0.107	0.090	0.089	0.119	0.075
BPR-MF	0.027	0.113	0.050	0.108	0.092	0.102	0.142	0.097
FPMC	0.015	0.078	0.028	0.073	0.050	0.068	0.077	0.064
FISM	0.009	0.046	0.015	0.042	0.027	0.038	0.040	0.035
FOSSIL	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 24: Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the RETAILROCKET dataset (sorted by P@20).

Algorithm	P@20	R@20	P@10	R@10	P@5	R@5	P@3	R@3
S-SKNN	0.057	0.480	0.096	0.434	0.156	0.376	0.214	0.326
S-KNN	0.056	0.478	0.095	0.433	0.152	0.373	0.208	0.323
V-SKNN	0.055	0.462	0.093	0.417	0.152	0.364	0.209	0.316
SMF	0.047	0.397	0.074	0.335	0.113	0.271	0.148	0.219
GRU4REC	0.046	0.400	0.073	0.339	0.111	0.271	0.143	0.215
AR	0.041	0.360	0.068	0.318	0.109	0.268	0.147	0.225
SR	0.038	0.342	0.067	0.313	0.110	0.269	0.149	0.227
MC	0.030	0.284	0.056	0.271	0.097	0.242	0.136	0.210
SF-SKNN	0.030	0.285	0.057	0.280	0.105	0.263	0.156	0.240
BPR-MF	0.029	0.286	0.056	0.282	0.107	0.277	0.172	0.272
IKNN	0.026	0.199	0.042	0.167	0.061	0.129	0.077	0.103
FPMC	0.023	0.253	0.043	0.246	0.083	0.239	0.133	0.233
FOSSIL	0.006	0.057	0.009	0.045	0.012	0.032	0.016	0.025
FISM	0.005	0.059	0.008	0.045	0.011	0.034	0.014	0.027

Table 25: Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the ZALANDO dataset (sorted by P@20).

Algorithm	P@20	R@20	P@10	R@10	P@5	R@5	P@3	R@3
V-SKNN	0.076	0.219	0.129	0.195	0.210	0.168	0.282	0.141
S-SKNN	0.075	0.217	0.127	0.192	0.198	0.159	0.252	0.127
S-KNN	0.074	0.202	0.115	0.169	0.164	0.130	0.196	0.098
GRU4REC	0.065	0.181	0.109	0.161	0.181	0.141	0.247	0.120
SMF	0.062	0.175	0.103	0.154	0.169	0.133	0.228	0.112
AR	0.060	0.179	0.105	0.161	0.178	0.141	0.241	0.118
SR	0.060	0.174	0.106	0.161	0.185	0.146	0.260	0.127
MC	0.054	0.167	0.100	0.157	0.176	0.142	0.246	0.123
SF-SKNN	0.053	0.165	0.101	0.160	0.182	0.147	0.254	0.129
IKNN	0.045	0.134	0.080	0.122	0.137	0.109	0.196	0.096
BPR-MF	0.026	0.089	0.048	0.084	0.088	0.078	0.133	0.072
FPMC	0.016	0.060	0.029	0.055	0.051	0.051	0.078	0.047
FOSSIL	0.009	0.026	0.013	0.020	0.018	0.015	0.023	0.011
FISM	0.007	0.028	0.013	0.024	0.021	0.021	0.031	0.018

Table 26: Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the 8TRACKS dataset (sorted by P@20).

Algorithm	P@20	R@20	P@10	R@10	P@5	R@5	P@3	R@3
V-SKNN	0.0122	0.0308	0.0138	0.0184	0.0125	0.0084	0.0109	0.0047
S-KNN	0.0117	0.0313	0.0119	0.0171	0.0090	0.0067	0.0059	0.0026
S-SKNN	0.0100	0.0266	0.0114	0.0162	0.0097	0.0070	0.0065	0.0029
AR	0.0087	0.0219	0.0112	0.0147	0.0136	0.0090	0.0155	0.0062
SMF	0.0086	0.0218	0.0104	0.0135	0.0118	0.0078	0.0126	0.0052
IKNN	0.0060	0.0149	0.0077	0.0102	0.0092	0.0063	0.0103	0.0043
SR	0.0055	0.0140	0.0077	0.0099	0.0098	0.0067	0.0111	0.0046
GRU4REC	0.0037	0.0095	0.0046	0.0061	0.0056	0.0039	0.0064	0.0027
SF-SKNN	0.0032	0.0081	0.0052	0.0067	0.0079	0.0052	0.0104	0.0042
MC	0.0025	0.0064	0.0036	0.0048	0.0051	0.0035	0.0065	0.0028
FOSSIL	0.0021	0.0043	0.0024	0.0025	0.0027	0.0013	0.0029	0.0009
BPR-MF	0.0018	0.0037	0.0021	0.0020	0.0022	0.0011	0.0019	0.0006
FPMC	0.0005	0.0012	0.0008	0.0009	0.0012	0.0007	0.0016	0.0006
FISM	0.0004	0.0008	0.0004	0.0005	0.0004	0.0002	0.0005	0.0002

Table 27: Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the AOTM dataset (sorted by P@20).

Algorithm	P@20	R@20	P@10	R@10	P@5	R@5	P@3	R@3
S-KNN	0.0155	0.0440	0.0157	0.0214	0.0099	0.0058	0.0056	0.0023
V-SKNN	0.0133	0.0361	0.0145	0.0196	0.0118	0.0078	0.0106	0.0056
S-SKNN	0.0125	0.0353	0.0122	0.0178	0.0084	0.0065	0.0054	0.0027
SMF	0.0084	0.0259	0.0105	0.0163	0.0130	0.0104	0.0143	0.0070
AR	0.0066	0.0183	0.0082	0.0119	0.0095	0.0068	0.0107	0.0049
IKNN	0.0056	0.0155	0.0069	0.0102	0.0082	0.0062	0.0090	0.0043
SR	0.0053	0.0146	0.0070	0.0098	0.0082	0.0061	0.0092	0.0041
SF-SKNN	0.0024	0.0074	0.0043	0.0068	0.0075	0.0062	0.0108	0.0055
MC	0.0022	0.0069	0.0034	0.0056	0.0049	0.0043	0.0060	0.0033
FOSSIL	0.0012	0.0036	0.0013	0.0023	0.0015	0.0013	0.0017	0.0008
GRU4REC	0.0010	0.0027	0.0011	0.0015	0.0014	0.0008	0.0015	0.0005
BPR-MF	0.0004	0.0018	0.0005	0.0012	0.0005	0.0007	0.0005	0.0005
FISM	0.0004	0.0013	0.0005	0.0007	0.0005	0.0004	0.0004	0.0002
FPMC	0.0002	0.0007	0.0003	0.0006	0.0004	0.0004	0.0005	0.0004

Table 28: Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the 30MUSIC dataset (sorted by P@20).

Algorithm	P@20	R@20	P@10	R@10	P@5	R@5	P@3	R@3
V-SKNN	0.1117	0.2438	0.1585	0.1948	0.1947	0.1371	0.2239	0.1044
S-SKNN	0.1110	0.2353	0.1439	0.1672	0.1431	0.0832	0.1344	0.0458
S-KNN	0.1035	0.2140	0.1295	0.1462	0.1283	0.0722	0.1216	0.0402
IKNN	0.0935	0.2023	0.1336	0.1611	0.1529	0.1015	0.1585	0.0651
AR	0.0914	0.1923	0.1244	0.1435	0.1354	0.0825	0.1386	0.0514
SR	0.0878	0.2010	0.1393	0.1750	0.1884	0.1353	0.2204	0.1042
SMF	0.0746	0.1655	0.1025	0.1272	0.1290	0.0876	0.1451	0.0626
GRU4REC	0.0404	0.0988	0.0627	0.0856	0.0932	0.0715	0.1236	0.0611
SF-SKNN	0.0319	0.0865	0.0591	0.0852	0.1027	0.0793	0.1447	0.0702
MC	0.0313	0.0852	0.0553	0.0811	0.0928	0.0743	0.1333	0.0679
BPR-MF	0.0172	0.0340	0.0290	0.0292	0.0442	0.0227	0.0569	0.0185
FOSSIL	0.0123	0.0188	0.0134	0.0117	0.0134	0.0055	0.0099	0.0027
FPMC	0.0046	0.0146	0.0079	0.0126	0.0137	0.0110	0.0205	0.0100
FISM	0.0015	0.0036	0.0019	0.0022	0.0025	0.0014	0.0025	0.0009

Table 29: Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the NOWPLAYING dataset (sorted by P@20).

Algorithm	P@20	R@20	P@10	R@10	P@5	R@5	P@3	R@3
S-SKNN	0.0726	0.1944	0.0890	0.1296	0.0950	0.0694	0.0935	0.0405
V-SKNN	0.0718	0.1909	0.0900	0.1303	0.1048	0.0873	0.1169	0.0633
S-KNN	0.0680	0.1824	0.0841	0.1186	0.0868	0.0622	0.0890	0.0362
AR	0.0554	0.1551	0.0724	0.1086	0.0876	0.0705	0.0968	0.0491
SR	0.0501	0.1465	0.0717	0.1132	0.0945	0.0826	0.1080	0.0614
SMF	0.0499	0.1453	0.0668	0.1043	0.0840	0.0709	0.0966	0.0525
IKNN	0.0492	0.1385	0.0639	0.0974	0.0755	0.0608	0.0809	0.0405
SF-SKNN	0.0280	0.0903	0.0495	0.0816	0.0761	0.0660	0.0992	0.0539
GRU4REC	0.0272	0.0810	0.0383	0.0601	0.0523	0.0444	0.0636	0.0343
MC	0.0250	0.0845	0.0415	0.0724	0.0625	0.0573	0.0810	0.0473
FOSSIL	0.0169	0.0412	0.0229	0.0291	0.0308	0.0204	0.0359	0.0149
BPR-MF	0.0156	0.0393	0.0231	0.0328	0.0358	0.0275	0.0492	0.0240
FPMC	0.0061	0.0244	0.0102	0.0211	0.0171	0.0181	0.0248	0.0161
FISM	0.0023	0.0077	0.0033	0.0064	0.0051	0.0053	0.0072	0.0044

Table 30: Precision (P) and Recall (R) results for a list length of 20, 10, 5, and 3 on the CLEF dataset (sorted by P@20).

Algorithm	P@20	R@20	P@10	R@10	P@5	R@5	P@3	R@3
GRU4REC	0.072	0.626	0.100	0.454	0.128	0.298	0.144	0.204
V-SKNN	0.069	0.593	0.089	0.413	0.108	0.262	0.119	0.180
S-SKNN	0.066	0.579	0.086	0.404	0.097	0.244	0.101	0.154
S-KNN	0.066	0.577	0.085	0.399	0.096	0.241	0.099	0.152
SF-SKNN	0.064	0.565	0.082	0.390	0.095	0.241	0.098	0.151
SMF	0.062	0.527	0.084	0.377	0.107	0.246	0.120	0.167
FPMC	0.060	0.515	0.078	0.347	0.093	0.216	0.110	0.154
MC	0.059	0.510	0.081	0.368	0.107	0.251	0.119	0.170
FOSSIL	0.059	0.504	0.075	0.334	0.090	0.205	0.103	0.143
AR	0.058	0.506	0.080	0.364	0.101	0.239	0.115	0.170
SR	0.058	0.502	0.081	0.366	0.108	0.251	0.115	0.162
FISM	0.058	0.506	0.077	0.357	0.095	0.227	0.105	0.156
IKNN	0.050	0.418	0.065	0.290	0.086	0.197	0.099	0.141
BPR-MF	0.016	0.147	0.024	0.120	0.042	0.105	0.062	0.094

D Additional Single Split Results

Table 31: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the TMALL dataset with a single split (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
S-SKNN	0.181	0.385	0.560	0.019	0.177	0.329	0.375	0.021
S-KNN	0.177	0.398	0.461	0.020	0.173	0.334	0.306	0.022
V-SKNN	0.169	0.353	0.570	0.019	0.166	0.298	0.405	0.021
SF-SKNN	0.141	0.237	0.577	0.015	0.140	0.220	0.441	0.018
SMF	0.140	0.298	0.461	0.023	0.136	0.245	0.341	0.023
AR	0.131	0.254	0.628	0.019	0.128	0.214	0.457	0.021
SR	0.131	0.243	0.683	0.019	0.129	0.209	0.507	0.020
GRU4REC	0.123	0.263	0.171	0.029	0.120	0.213	0.117	0.032
MC	0.123	0.214	0.673	0.018	0.121	0.188	0.516	0.019
IKNN	0.049	0.147	0.801	0.006	0.047	0.111	0.644	0.006
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
S-SKNN	0.168	0.265	0.240	0.023	0.157	0.215	0.112	0.112
S-KNN	0.163	0.263	0.199	0.024	0.151	0.209	0.106	0.106
V-SKNN	0.158	0.241	0.282	0.023	0.148	0.199	0.109	0.109
SF-SKNN	0.136	0.194	0.309	0.020	0.131	0.168	0.101	0.101
SMF	0.129	0.195	0.244	0.024	0.122	0.160	0.092	0.092
AR	0.123	0.174	0.312	0.023	0.117	0.147	0.094	0.094
SR	0.124	0.173	0.348	0.021	0.118	0.148	0.095	0.095
GRU4REC	0.114	0.169	0.082	0.034	0.107	0.139	0.083	0.083
MC	0.117	0.160	0.361	0.021	0.113	0.139	0.092	0.092
IKNN	0.042	0.077	0.471	0.007	0.038	0.056	0.024	0.024

Table 32: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the RETAILROCKET dataset with a single split (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
S-SKNN	0.333	0.580	0.272	0.051	0.329	0.524	0.170	0.060
S-KNN	0.332	0.571	0.250	0.052	0.329	0.520	0.155	0.061
V-SKNN	0.327	0.582	0.272	0.061	0.323	0.520	0.172	0.069
SF-SKNN	0.301	0.463	0.224	0.036	0.299	0.444	0.168	0.049
SR	0.270	0.504	0.299	0.047	0.267	0.452	0.201	0.054
SMF	0.270	0.557	0.313	0.056	0.264	0.479	0.198	0.061
AR	0.265	0.485	0.286	0.058	0.261	0.425	0.189	0.064
MC	0.261	0.468	0.250	0.040	0.258	0.425	0.184	0.049
GRU4REC	0.260	0.559	0.291	0.055	0.254	0.475	0.187	0.062
IKNN	0.121	0.284	0.334	0.031	0.118	0.238	0.219	0.036
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
S-SKNN	0.320	0.456	0.098	0.069	0.305	0.391	0.236	0.236
S-KNN	0.320	0.453	0.090	0.070	0.305	0.386	0.242	0.242
V-SKNN	0.313	0.446	0.104	0.075	0.298	0.383	0.230	0.230
SF-SKNN	0.293	0.399	0.110	0.060	0.281	0.346	0.230	0.230
SR	0.257	0.379	0.123	0.062	0.244	0.322	0.184	0.184
SMF	0.252	0.388	0.119	0.067	0.234	0.309	0.177	0.177
AR	0.253	0.366	0.115	0.074	0.240	0.310	0.187	0.187
MC	0.250	0.366	0.121	0.059	0.236	0.308	0.180	0.180
GRU4REC	0.240	0.377	0.114	0.069	0.223	0.300	0.164	0.164
IKNN	0.111	0.185	0.131	0.041	0.103	0.154	0.067	0.067

Table 33: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the ZALANDO dataset with a single split (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
SR	0.306	0.498	0.525	0.059	0.304	0.473	0.381	0.060
MC	0.304	0.469	0.504	0.052	0.302	0.452	0.381	0.055
IKNN	0.271	0.410	0.597	0.033	0.269	0.388	0.432	0.036
GRU4REC	0.267	0.483	0.290	0.073	0.265	0.442	0.223	0.074
AR	0.265	0.483	0.431	0.073	0.262	0.450	0.311	0.073
SMF	0.253	0.463	0.329	0.075	0.250	0.422	0.248	0.076
SF-SKNN	0.251	0.451	0.419	0.046	0.250	0.440	0.323	0.053
V-SKNN	0.237	0.517	0.396	0.077	0.234	0.478	0.275	0.076
S-SKNN	0.224	0.510	0.395	0.066	0.221	0.464	0.257	0.068
S-KNN	0.181	0.461	0.301	0.069	0.176	0.392	0.197	0.071
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
SR	0.299	0.435	0.256	0.058	0.287	0.384	0.210	0.210
MC	0.298	0.422	0.262	0.056	0.288	0.379	0.216	0.216
IKNN	0.266	0.363	0.287	0.040	0.258	0.330	0.199	0.199
GRU4REC	0.258	0.394	0.167	0.073	0.245	0.336	0.174	0.174
AR	0.256	0.404	0.216	0.069	0.243	0.347	0.163	0.163
SMF	0.243	0.372	0.182	0.074	0.230	0.317	0.163	0.163
SF-SKNN	0.245	0.406	0.227	0.057	0.231	0.345	0.143	0.143
V-SKNN	0.226	0.419	0.185	0.071	0.208	0.343	0.103	0.103
S-SKNN	0.211	0.391	0.161	0.070	0.193	0.311	0.100	0.100
S-KNN	0.164	0.300	0.124	0.074	0.147	0.226	0.087	0.087

Table 34: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the 8TRACKS dataset with a single split (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
AR	0.0135	0.0410	0.7769	0.0399	0.0126	0.0276	0.5772	0.0545
SR	0.0123	0.0329	0.8875	0.0243	0.0116	0.0233	0.7261	0.0302
SMF	0.0115	0.0476	0.0772	0.1197	0.0102	0.0290	0.0556	0.1303
V-SKNN	0.0110	0.0490	0.7180	0.0290	0.0098	0.0313	0.5317	0.0322
MC	0.0101	0.0234	0.8365	0.0152	0.0098	0.0179	0.7050	0.0179
S-KNN	0.0098	0.0438	0.6122	0.0272	0.0086	0.0267	0.4343	0.0298
S-SKNN	0.0097	0.0402	0.8543	0.0197	0.0087	0.0265	0.6465	0.0238
GRU4REC	0.0095	0.0376	0.0593	0.1930	0.0085	0.0231	0.0445	0.2140
SF-SKNN	0.0089	0.0217	0.7713	0.0157	0.0086	0.0171	0.6555	0.0219
IKNN	0.0072	0.0251	0.9852	0.0063	0.0066	0.0165	0.8756	0.0069
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
AR	0.0114	0.0185	0.3933	0.0774	0.0104	0.0140	0.0078	0.0078
SR	0.0107	0.0166	0.5237	0.0374	0.0099	0.0128	0.0078	0.0078
SMF	0.0087	0.0172	0.0406	0.1401	0.0073	0.0113	0.0044	0.0044
V-SKNN	0.0079	0.0167	0.4041	0.0324	0.0063	0.0097	0.0040	0.0040
MC	0.0092	0.0134	0.5477	0.0216	0.0086	0.0107	0.0070	0.0070
S-KNN	0.0068	0.0128	0.3056	0.0267	0.0055	0.0074	0.0044	0.0044
S-SKNN	0.0070	0.0137	0.3984	0.0256	0.0057	0.0079	0.0043	0.0043
GRU4REC	0.0073	0.0140	0.0336	0.2350	0.0062	0.0093	0.0040	0.0040
SF-SKNN	0.0080	0.0131	0.5072	0.0286	0.0074	0.0101	0.0052	0.0052
IKNN	0.0058	0.0107	0.6635	0.0075	0.0051	0.0075	0.0034	0.0034

Table 35: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the AOTM dataset with a single split (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
SF-SKNN	0.0275	0.0440	0.8591	0.0828	0.0273	0.0403	0.7673	0.1025
SMF	0.0204	0.0468	0.9262	0.0941	0.0197	0.0361	0.8294	0.0941
GRU4REC	0.0154	0.0427	0.6523	0.1665	0.0146	0.0312	0.5081	0.1759
SR	0.0152	0.0449	0.9439	0.1061	0.0143	0.0318	0.8140	0.1143
MC	0.0134	0.0348	0.8996	0.0813	0.0128	0.0262	0.7889	0.0902
AR	0.0119	0.0426	0.8523	0.1409	0.0109	0.0283	0.6723	0.1536
V-SKNN	0.0104	0.0721	0.7971	0.1567	0.0083	0.0415	0.6049	0.1662
IKNN	0.0100	0.0384	0.9854	0.0482	0.0090	0.0242	0.8660	0.0490
S-SKNN	0.0095	0.0737	0.8917	0.1192	0.0071	0.0406	0.6652	0.1322
S-KNN	0.0087	0.0740	0.6400	0.1414	0.0059	0.0345	0.4599	0.1416
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
SF-SKNN	0.0267	0.0360	0.6205	0.1192	0.0257	0.0315	0.0212	0.0212
SMF	0.0186	0.0280	0.6845	0.0927	0.0175	0.0232	0.0132	0.0132
GRU4REC	0.0134	0.0225	0.3629	0.1840	0.0121	0.0166	0.0087	0.0087
SR	0.0130	0.0217	0.6154	0.1226	0.0118	0.0167	0.0083	0.0083
MC	0.0119	0.0198	0.6326	0.0985	0.0109	0.0151	0.0078	0.0078
AR	0.0095	0.0178	0.4853	0.1689	0.0083	0.0127	0.0052	0.0052
V-SKNN	0.0051	0.0174	0.4490	0.1646	0.0027	0.0067	0.0001	0.0001
IKNN	0.0079	0.0153	0.6629	0.0499	0.0067	0.0103	0.0040	0.0040
S-SKNN	0.0035	0.0124	0.4265	0.1392	0.0016	0.0040	0.0000	0.0000
S-KNN	0.0028	0.0096	0.3142	0.1387	0.0014	0.0033	0.0001	0.0001

Table 36: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the 30MUSIC dataset with a single split (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
SR	0.2690	0.3744	0.5904	0.0373	0.2672	0.3499	0.4619	0.0389
MC	0.2653	0.3302	0.4001	0.0283	0.2646	0.3203	0.3455	0.0340
GRU4REC	0.2354	0.3651	0.4029	0.0665	0.2334	0.3372	0.3077	0.0669
SMF	0.2145	0.3614	0.3974	0.0608	0.2121	0.3275	0.3013	0.0605
SF-SKNN	0.2123	0.3320	0.3404	0.0284	0.2118	0.3258	0.2929	0.0366
IKNN	0.1352	0.3412	0.6758	0.0219	0.1319	0.2943	0.5043	0.0234
V-SKNN	0.1192	0.4134	0.4384	0.0589	0.1130	0.3263	0.2911	0.0599
AR	0.1157	0.3518	0.5352	0.0437	0.1107	0.2810	0.3886	0.0456
S-SKNN	0.1153	0.4119	0.4195	0.0544	0.1087	0.3190	0.2681	0.0593
S-KNN	0.0938	0.3609	0.2848	0.0595	0.0869	0.2626	0.1799	0.0658
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
SR	0.2629	0.3179	0.3270	0.0387	0.2572	0.2930	0.2282	0.2282
MC	0.2625	0.3051	0.2790	0.0374	0.2590	0.2900	0.2332	0.2332
GRU4REC	0.2286	0.3011	0.2289	0.0653	0.2212	0.2688	0.1830	0.1830
SMF	0.2064	0.2851	0.2200	0.0587	0.1981	0.2488	0.1583	0.1583
SF-SKNN	0.2083	0.3013	0.2364	0.0432	0.1987	0.2598	0.1507	0.1507
IKNN	0.1215	0.2176	0.3427	0.0227	0.1084	0.1600	0.0700	0.0700
V-SKNN	0.0958	0.1972	0.1927	0.0630	0.0785	0.1202	0.0494	0.0494
AR	0.0985	0.1905	0.2579	0.0493	0.0849	0.1304	0.0519	0.0519
S-SKNN	0.0913	0.1881	0.1700	0.0660	0.0745	0.1135	0.0471	0.0471
S-KNN	0.0719	0.1481	0.1129	0.0742	0.0589	0.0904	0.0367	0.0367

Table 37: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the NOWPLAYING dataset with a single split (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
SR	0.0856	0.1825	0.4629	0.0309	0.0831	0.1466	0.3390	0.0346
MC	0.0813	0.1474	0.3576	0.0248	0.0798	0.1255	0.2833	0.0287
SF-SKNN	0.0787	0.1602	0.3015	0.0264	0.0774	0.1431	0.2383	0.0333
SMF	0.0782	0.1881	0.3560	0.0322	0.0753	0.1454	0.2585	0.0332
GRU4REC	0.0771	0.1792	0.2202	0.0523	0.0742	0.1370	0.1647	0.0568
V-SKNN	0.0670	0.2291	0.3358	0.0445	0.0624	0.1627	0.2248	0.0497
S-SKNN	0.0669	0.2406	0.3436	0.0407	0.0618	0.1674	0.2186	0.0468
AR	0.0647	0.1866	0.4137	0.0381	0.0613	0.1378	0.2869	0.0441
S-KNN	0.0604	0.2241	0.2312	0.0453	0.0554	0.1512	0.1487	0.0509
IKNN	0.0467	0.1554	0.5526	0.0144	0.0437	0.1120	0.3960	0.0145
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
SR	0.0789	0.1146	0.2290	0.0390	0.0744	0.0949	0.0584	0.0584
MC	0.0769	0.1041	0.2090	0.0336	0.0738	0.0903	0.0610	0.0610
SF-SKNN	0.0737	0.1157	0.1743	0.0403	0.0682	0.0914	0.0500	0.0500
SMF	0.0705	0.1085	0.1799	0.0341	0.0658	0.0879	0.0491	0.0491
GRU4REC	0.0700	0.1053	0.1196	0.0604	0.0651	0.0839	0.0510	0.0510
V-SKNN	0.0543	0.1019	0.1465	0.0543	0.0466	0.0677	0.0316	0.0316
S-SKNN	0.0532	0.1017	0.1329	0.0523	0.0450	0.0655	0.0306	0.0306
AR	0.0555	0.0947	0.1830	0.0531	0.0499	0.0701	0.0350	0.0350
S-KNN	0.0472	0.0888	0.0936	0.0560	0.0402	0.0578	0.0282	0.0282
IKNN	0.0384	0.0724	0.2597	0.0141	0.0332	0.0494	0.0214	0.0214

Table 38: Hit rate (HR), Mean reciprocal rank (MRR), item coverage (COV), and average popularity (POP) results for a list length of 20, 10, 5, 3, and 1 on the CLEF dataset with a single split (sorted by MRR@20).

Algorithm	MRR@20	HR@20	COV@20	POP@20	MRR@10	HR@10	COV@10	POP@10
GRU4REC	0.253	0.724	0.154	0.051	0.242	0.564	0.123	0.064
SR	0.213	0.623	0.754	0.055	0.202	0.451	0.715	0.084
MC	0.213	0.625	0.755	0.056	0.202	0.459	0.724	0.080
SMF	0.207	0.646	0.765	0.042	0.194	0.472	0.709	0.042
V-SKNN	0.197	0.683	0.756	0.053	0.183	0.487	0.682	0.080
S-SKNN	0.190	0.670	0.764	0.052	0.175	0.463	0.682	0.081
S-KNN	0.186	0.661	0.760	0.052	0.172	0.453	0.675	0.083
SF-SKNN	0.179	0.636	0.750	0.052	0.165	0.433	0.680	0.082
AR	0.178	0.631	0.733	0.058	0.164	0.435	0.653	0.103
IKNN	0.158	0.510	0.796	0.009	0.148	0.363	0.757	0.010
Algorithm	MRR@5	HR@5	COV@5	POP@5	MRR@3	HR@3	MRR@1	HR@1
GRU4REC	0.219	0.384	0.101	0.088	0.195	0.280	0.132	0.132
SR	0.183	0.308	0.636	0.110	0.167	0.238	0.113	0.113
MC	0.184	0.325	0.652	0.110	0.165	0.241	0.107	0.107
SMF	0.172	0.300	0.639	0.023	0.155	0.225	0.104	0.104
V-SKNN	0.160	0.316	0.585	0.082	0.137	0.215	0.081	0.081
S-SKNN	0.154	0.304	0.595	0.101	0.133	0.208	0.077	0.077
S-KNN	0.151	0.298	0.592	0.103	0.130	0.205	0.076	0.076
SF-SKNN	0.145	0.284	0.593	0.104	0.126	0.199	0.072	0.072
AR	0.143	0.278	0.517	0.170	0.125	0.198	0.070	0.070
IKNN	0.131	0.242	0.673	0.010	0.117	0.179	0.073	0.073

Performance Comparison of Neural and Non-Neural Approaches to Session-based Recommendation

Malte Ludewig
TU Dortmund, Germany
malte.ludewig@tu-dortmund.de

Noemi Mauro
University of Torino, Italy
noemi.mauro@unito.it

Sara Latifi
University of Klagenfurt, Austria
sara.latifi@aau.at

Dietmar Jannach
University of Klagenfurt, Austria
dietmar.jannach@aau.at

ABSTRACT

The benefits of neural approaches are undisputed in many application areas. However, today's research practice in applied machine learning—where researchers often use a variety of baselines, datasets, and evaluation procedures—can make it difficult to understand how much progress is actually achieved through novel technical approaches. In this work, we focus on the fast-developing area of session-based recommendation and aim to contribute to a better understanding of what represents the state-of-the-art.

To that purpose, we have conducted an extensive set of experiments, using a variety of datasets, in which we benchmarked four neural approaches that were published in the last three years against each other and against a set of simpler baseline techniques, e.g., based on nearest neighbors. The evaluation of the algorithms under the exact same conditions revealed that the benefits of applying today's neural approaches to session-based recommendations are still limited. In the majority of the cases, and in particular when precision and recall are used, it turned out that simple techniques in most cases outperform recent neural approaches. Our findings therefore point to certain major limitations of today's research practice. By sharing our evaluation framework publicly, we hope that some of these limitations can be overcome in the future.

CCS CONCEPTS

• **Information systems → Recommender systems**.

KEYWORDS

Session-based Recommendation; Evaluation; Reproducibility

ACM Reference Format:
Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. 2019. Performance Comparison of Neural and Non-Neural Approaches to Session-based Recommendation. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3298689.3347041

RecSys '19, September 16–20, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-6243-6/19/09…$15.00
https://doi.org/10.1145/3298689.3347041

1 INTRODUCTION

In recent years, we could observe an increased research interest in *session-based* recommendation problems. In such settings, the problem is not to make relevance predictions for items given the users' long-term preferences, but to make recommendations given only a few user interactions in an ongoing session [19]. While such scenarios have been addressed in the literature previously, e.g., for web usage prediction [18], they have recently received more attention, e.g., due to the availability of public datasets.

From a technical perspective, almost all session-based algorithms proposed in recent years are based on deep learning ("neural") architectures. A landmark work in this area is the GRU4REC method, which is based on Recurrent Neural Networks (RNNs) [4, 5]. Today, GRU4REC is often used as a baseline algorithm in experimental evaluations. However, recent research [7, 15] indicates that simpler methods based on nearest-neighbor techniques can outperform GRU4REC in terms of certain accuracy measures. Therefore, when new neural algorithms are published and benchmarked against GRU4REC alone, it is not clear whether or not these new methods are actually leading to progress beyond the more simple techniques.

This problem of unclear progress in applied machine learning is not entirely new. In the information retrieval (IR) field, for example, researchers already found in 2009 that the improvements reported over the years "don't add up" [1]. Recent analyses [10, 16] furthermore indicate that some neural approaches that were recently published at top conferences do not outperform long-established baseline methods, when these are well tuned. The reasons for this non-progress lie in the choice of the baselines used in the experimental evaluations or the limited efforts by the authors to fine-tune the baselines. Sometimes, another problem is the lack of reproducibility of the results. Today, publishing the code of the algorithms is more and more encouraged. However, often the code used for data pre-processing, data splitting, hyper-parameter optimization, and evaluating is not provided. Given that many of these implementation details can affect accuracy, it is often very challenging to make reliable conclusions.

With this work, our goal is to shed light on the progress in the area of session-based recommendation algorithms. We report the results of an in-depth comparison of four recent neural algorithms and a set of mostly simpler baseline algorithms. All algorithms were benchmarked under identical settings within an evaluation framework that we built upon the code from [5]. Our results indicate that the progress that is achieved with neural approaches is sometimes

very limited, and that well-tuned baselines often outperform even the latest complex models.

Generally, these observations call for improved research practices, as discussed previously in [12]. The availability of an evaluation environment for reproducible research can be one piece of this puzzle. We therefore publicly share our evaluation framework, which includes also code for data splitting, hyper-parameter optimization and a number of additional metrics.

2 BENCHMARKED ALGORITHMS

We have considered the four neural approaches shown in Table 1 in our comparison. We selected them by systematically scanning the proceedings of top-ranked conference series of the last three years. We only included works for which the source code was available and which did not use side information.

Table 1: Neural Recommendation Strategies

GRU4REC (ICLR'16, CIKM'18)	GRU4REC [5] was the first neural approach that employed RNNs for session-based recommendation. The technique was later on improved using more effective loss functions [4].
NARM (CIKM'17)	This model extends GRU4REC and improves its session modeling with the introduction of an attention mechanism. This also proved to be advantageous in the NLP field [8].
STAMP (KDD'18)	In contrast to NARM, this model does not rely on an RNN anymore. Instead, the session is modeled solely with an attention mechanism in order to improve both effectiveness and efficiency [13].
NEXTITNET (WSDM'19)	This recent model also discards RNNs to model user sessions. In contrast to STAMP, convolutional neural networks are adopted with a few domain-specific enhancements [21].

As baselines we use the five techniques that were also used in [15], as well as a recent, more complex approach based on context trees (CT) [17]. All baselines methods shown in Table 2 have the advantage that they can take new interactions immediately into account without retraining, and they only have a small set of parameters to tune. Furthermore, scalability can be ensured for the neighborhood-based techniques through adequate sampling as discussed in [7]. We initially considered additional neural approaches such as [2, 9, 11, 14], but we did not include them in our evaluation for different reasons, e.g., because the source code was not available, or the algorithm also uses side information. We also did not consider *sequential* approaches like [3, 6, 20], because they are not really designed for session-based scenarios or require user IDs in the datasets.

Table 2: Baseline Strategies

AR	Learns and applies association rules of size two. Works by simply counting pairwise item co-occurrences in the training sessions.
SR	Similar to AR, but learns *sequential* rules of size two, i.e., it counts how often one item appeared after another (possibly with elements in between) in the training sessions.
S-KNN	A session-based nearest-neighbor technique. Every item in the session is assumed to be equally important when computing similarities.
VS-KNN	Like S-KNN, but uses a similarity function that puts more emphasis on the more recent events in a session.
CT	This technique is based on context trees. It is non-parametric and showed promising results in [17].

3 DATASETS AND EVALUATION APPROACH

3.1 Datasets

We conducted experiments with seven datasets, four from the e-commerce domain and three from the music domain, see Table 3. Six of these datasets are publicly available. These datasets were also used for the comparison of algorithms in [8, 15] and [13].

Table 3: Datasets

RSC15	E-commerce dataset used in the 2015 ACM RecSys Challenge.
RETAIL	An e-commerce dataset from the company Retail Rocket.
DIGI	An e-commerce dataset shared by the company Diginetica.
ZALANDO	A non-public dataset consisting of interaction logs from the European fashion retailer Zalando.
30MU	Music listening logs obtained from Last.fm.
NOWP	Music listening logs obtained from Twitter.
AOTM	A public music dataset containing music playlists.

Some previous works on session-based recommendation use a single training-test split in their evaluation or very small subsets of the original datasets (e.g., only 1/64 of the RSC15 dataset) [4, 5, 8, 13]. In our work, we followed the approach of [15] and created, for each dataset, five subsets contiguous in time to be able to make multiple measurements in order to minimize the risk of random effects. Table 4 shows the average characteristics of these multiple subsets. Pointers to the resulting datasets and the train-test splits used in the experiments can be found online[1], together with the code of our evaluation framework. For all datasets, we removed sessions that contained only one interaction.

Table 4: Characteristics of the datasets. The values are averaged over all five splits.

Dataset	RSC15	RETAIL	DIGI	ZALANDO	30MU	NOWP	AOTM
Actions	5.4M	210k	264k	4.5M	640k	271k	307k
Sessions	1.4M	60k	55k	365k	37k	27k	22k
Items	29k	32k	32k	189k	91k	75k	91k
Days covered	31	27	31	90	90	90	90
Actions/Session	3.95	3.54	4.78	12.43	17.11	10.04	14.02
Items/Session	3.17	2.56	4.01	8.39	14.47	9.38	14.01
Actions/Day	175k	8k	8.5k	50k	7k	3.0k	3.4k
Sessions/Day	44k	2.2k	1.7k	4k	300	243	243

3.2 Experimental Procedure

Hyper-Parameter Optimization. We tuned the hyper-parameters for all methods for each dataset systematically, using a subset of the training data—covering the same amount of days as the test set—for validation. As the training process can be time-consuming and the parameter space is large, we applied a random optimization approach with 100 iterations as in [4, 8, 13] (50 iterations for NARM) to find a suitable set of parameters. All models were optimized for the Mean Reciprocal Rank (MRR@20). The ranges and the final values of the hyper-parameters for each dataset can be found online.

Protocol and Metrics. Similar to [4, 5] and other works, we used the last n days of each dataset as test data and the rest for training. For each session in the test data, we incrementally "revealed" one interaction after the other. After each revealed interaction, we

[1] https://rn5l.github.io/session-rec/index.html

computed recommendation lists and then compared the recommendations with the still hidden elements in the session.

In [5], where GRU4REC was proposed, and in subsequent works, the evaluation procedure is based on measuring to what extent an algorithm is able to predict the *immediate next* item in a session. Their corresponding measurement of the Hit Rate (HR@20) and the MRR@20 is therefore based on the existence of this next item in a given top-n recommendation list. In reality, however, usually more than one item is shown and being able to identify more than one relevant item for a given session is typically favorable over just predicting the immediate next one correctly. In this work, we therefore focus on traditional precision, recall, and mean average precision (MAP) measures, which consider all items that appear in the currently hidden part of the session as relevant. As the neural approaches are not explicitly designed to predict multiple items and for the sake of completeness, we report both types of measurements.

4 RESULTS

E-Commerce Domain. Table 5 shows the results for the domain of e-commerce.[2] On the RETAIL and the DIGI dataset, the nearest neighbor methods led to the highest accuracy results—averaged across folds—on all measures. For the ZALANDO dataset, neighborhood methods were again best, except for the MRR. The differences to the best complex model are in many cases significant.

Only for the RSC15 dataset we can observe that a neural method (NARM) is able to consistently outperform our best baseline VS-KNN on all measures. Interestingly, however, it is one of the earlier neural methods in this comparison. The results for the RSC15 dataset are generally different from the other results. The CT method, for example, was very competitive on the MRR for this dataset. STAMP, while being a very recent method, was not among the top performers except for this dataset. Given these observations, it seems that the RSC15 dataset has some unique characteristics that are different from the other e-commerce datasets.

For the larger ZALANDO and RSC15 datasets, we do not include measurements for the most recent NEXTITNET method. We found that the method does not scale well and we could not complete the hyper-parameter tuning process within weeks on our machines (also for two music datasets).

Music Domain. Table 6 shows the results for the music domain. The results are mostly aligned with the e-commerce results. On all datasets, the nearest-neighbor methods outperform all other techniques on precision, recall, MAP, and the hit rate. In terms of the MRR measure, the non-neural CT method consistently leads to the highest values. The simple SR method is again competitive in terms of the MRR, and GRU4REC as well as NARM are again among the top-performing neural approaches. The neighborhood methods in all cases are not in the leading positions in terms of the MRR and even lead to the lowest MRR performance on the AOTM dataset. The STAMP method can consistently be found at the lower ranks in this comparison.

[2]The highest value across all techniques is printed in bold; the highest value obtained by the other family of algorithms—baseline or complex model—is underlined. Stars indicate significant differences according to a Student's t-test with Bonferroni correction between the best-performing techniques from each category. $*: p<0.05$, $**: p<0.01$.

Table 5: Results for e-commerce datasets. The best values obtained for complex models and baselines are highlighted.[2]

Metrics	MAP@20	P@20	R@20	HR@20	MRR@20
RETAIL					
S-KNN	**0.0283**	**0.0532**	**0.4707**	**0.5788**	0.3370
VS-KNN	0.0278	0.0531	0.4632	0.5745	**0.3395**
GRU4REC	0.0272	0.0502	0.4559	0.5669	0.3237
NARM	0.0239	0.0440	0.4072	0.5549	0.3196
STAMP	0.0229	0.0428	0.3922	0.4620	0.2527
AR	0.0205	0.0387	0.3533	0.4367	0.2407
SR	0.0194	0.0362	0.3359	0.4174	0.2453
NEXTITNET	0.0173	0.0320	0.3051	0.3779	0.2038
CT	0.0162	0.0308	0.2902	0.3632	0.2305
DIGI					
S-KNN	**0.0255	*0.0596	**0.3715	*0.4748	0.1714
VS-KNN	0.0249	0.0584	0.3668	0.4729	**0.1784
GRU4REC	0.0247	0.0577	0.3617	0.4639	0.1644
NARM	0.0218	0.0528	0.3254	0.4188	0.1392
STAMP	0.0201	0.0489	0.3040	0.3917	0.1314
AR	0.0189	0.0463	0.2872	0.3720	0.1280
SR	0.0164	0.0406	0.2517	0.3277	0.1216
NEXTITNET	0.0149	0.0380	0.2416	0.2922	0.1424
CT	0.0115	0.0294	0.1860	0.2494	0.1075
ZALANDO					
VS-KNN	**0.0158**	**0.0740**	**0.1956**	**0.5162**	0.2487
S-KNN	0.0157	0.0738	0.1891	0.4352	0.1724
NARM	0.0144	0.0692	0.1795	0.4598	0.2248
GRU4REC	0.0143	0.0666	0.1797	0.4925	**0.3069**
SR	0.0136	0.0638	0.1739	0.4824	0.3043
AR	0.0133	0.0631	0.1690	0.4665	0.2579
CT	0.0118	0.0564	0.1573	0.4561	0.2993
STAMP	0.0104	0.0515	0.1359	0.3687	0.2065
RSC15					
NARM	**0.0357	**0.0735	**0.5109	*0.6751	0.3047
STAMP	0.0344	0.0713	0.4979	0.6654	0.3033
VS-KNN	0.0341	0.0707	0.4937	0.6512	0.2872
GRU4REC	0.0334	0.0682	0.4837	0.6480	0.2826
SR	0.0332	0.0684	0.4853	0.6506	0.3010
AR	0.0325	0.0673	0.4760	0.6361	0.2894
S-KNN	0.0318	0.0657	0.4658	0.5996	0.2620
CT	0.0316	0.0654	0.4710	0.6359	**0.3072**

Summary of Accuracy Measurements. Overall, across the domains we can observe that only in one single case—when using the RSC15 dataset—a rather early complex model was able to outperform relatively simple baselines. In the large majority of the cases in particular the neighborhood-based methods are better than newer neural approaches in terms of precision, recall, MAP, the hit rate and, in two cases also in terms of the MRR. When considering only the immediate next item for evaluation, and when using the MRR, the ranking of the algorithm often changes compared to the other measures. No consistent pattern was, however, found in terms of this measurement across the domains and datasets.

Some of the more recent approaches like NEXTITNET or STAMP often performed worse than GRU4REC according to our evaluation. In the original papers, they won such a comparison, although with different data subsets and evaluation procedures as in [4]. In the end,

Table 6: Results for the music domain datasets

Metrics	MAP@20	P@20	R@20	HR@20	MRR@20
NOWP					
VS-KNN	**0.0193	**0.0664	**0.1828	*0.2534	0.0810
S-KNN	0.0186	0.0655	0.1809	0.2450	0.0687
AR	0.0166	0.0564	0.1544	0.2076	0.0710
SR	0.0133	0.0466	0.1366	0.2002	0.1052
NARM	0.0118	0.0463	0.1274	0.1849	0.0894
GRU4REC	0.0116	0.0449	0.1361	0.2261	0.1076
STAMP	0.0111	0.0455	0.1245	0.1919	0.0897
CT	0.0065	0.0287	0.0893	0.1679	**0.1094**
30MU					
VS-KNN	**0.0309	**0.1090	**0.2347	**0.3830	0.1162
S-KNN	0.0290	0.1073	0.2217	0.3443	0.0898
AR	0.0254	0.0886	0.1930	0.3088	0.0960
SR	0.0240	0.0816	0.1937	0.3327	0.2410
NARM	0.0155	0.0675	0.1486	0.2956	0.1945
GRU4REC	0.0150	0.0617	0.1529	0.3273	0.2369
STAMP	0.0093	0.0411	0.0875	0.1539	0.0819
CT	0.0058	0.0308	0.0885	0.2882	*0.2502
AOTM					
S-KNN	**0.0037	**0.0139	**0.0390	**0.0417	0.0054
VS-KNN	0.0032	0.0116	0.0312	0.0352	0.0057
AR	0.0018	0.0076	0.0200	0.0233	0.0059
SR	0.0010	0.0047	0.0134	0.0186	0.0074
NARM	0.0009	0.0050	0.0146	0.0202	0.0088
CT	0.0006	0.0043	0.0126	0.0191	**0.0111
NEXTITNET	0.0004	0.0024	0.0071	0.0139	0.0065
STAMP	0.0003	0.0020	0.0063	0.0128	0.0088
GRU4REC	0.0003	0.0020	0.0063	0.0130	0.0074

it seems that progress in neural session-based recommendation is still limited, and the various reported improvements over the landmark GRU4REC method are seemingly not enough to consistently outperform much simpler techniques.

4.1 Additional Observations

Scalability. Scalability can be an issue for some of the complex models, with GRU4REC being among the faster approaches. The authors of STAMP and NARM, for example, use only 1/4 or 1/64 of the RSC15 dataset in their own experiments. Similarly, the largest dataset used for the evaluation of NEXTITNET has about 2 million sessions, which is a fraction of the original RSC15 dataset.

We measured the runtimes of training and prediction for all methods in all experiments. As an example, we report the results for RSC15 and ZALANDO in terms of the training time for one split and the average time needed to generate a recommendation list[3].

Methods like SR or VS-KNN do not learn complex models. They only need some time to count co-occurrences or prepare data structures. Also, the CT technique can be efficiently initialized. Training GRU4REC on one data split on our hardware took less than an hour. STAMP needed only slightly more time than GRU4REC, but NARM was four times slower. Finally, the most recent convolutional NEXTITNET method seems to be limited in terms of practical applicability as it needs more than one day for training on a GPU even for datasets of modest size. When datasets are used that comprise a larger set of items, e.g., the one from Zalando, the performance differences are even more pronounced. The CT method is generally fast enough when predicting for the RSC15 dataset, but it slows down rapidly when the number of items increases.

[3]Times were measured on a workstation computer with an Intel Core i7-4790k processor and a Nvidia Geforce GTX 1080 Ti graphics card (Cuda 10.1/CuDNN 7.5).

Table 7: Running times

Algorithm	Training RSC15	Training ZALANDO	Predicting (ms) RSC15	Predicting (ms) ZALANDO
GRU4REC (on GPU)	0.89h	1.51h	8.81	30.06
STAMP (on GPU)	1.25h	7.61h	13.79	51.84
NARM (on GPU)	4.36h	12.99h	9.72	28.69
NEXTITNET (on GPU)	26.39h	–	8.98	–
SR (on CPU)	17.35s	21.37s	3.40	8.66
VS-KNN (on CPU)	10.71s	5.48s	16.42	26.00
CT (on CPU)	5.91m	2.10h	57.66	327.83

Coverage and Popularity Bias. Previous work has indicated that some methods, in particular the simpler ones, can have a tendency to recommend more popular items [15]. At the same time, some algorithms can focus their recommendations on a small set of items that are recommended to everyone, which can be undesired in certain domains and lead to limited personalization.

To identify such potential differences, we measured the popularity bias of each algorithm by averaging the min-max normalized popularity values of the recommended items in the top-20 recommendations. Furthermore, we determined the fraction of items that ever appeared in the generated top-20 recommendations (*coverage*).

The general tendencies across datasets are as follows. In terms of the popularity bias, CT is usually very different from the other methods, and it focuses much more on popular items. For the other methods, no clear ranking was found across datasets. In many cases, however, GRU4REC is among the methods that recommend the least popular (or: most novel) items. GRU4REC also often has the highest and STAMP the lowest coverage. VS-KNN is similar to the other neural approaches in terms of coverage.

5 CONCLUSIONS

Our work indicates that even though a number of papers on session-based recommendations were published at very competitive conferences in the last years, progress seems to be still limited (or only *phantom progress*) despite the increasing computational complexity of the models. Similar to the IR domain, one main problem seems to lie in the choice of the baselines, and our work points to a potentially major limitation of today's research practice.

A general phenomenon in that context is that previous non-neural approaches—as well as simpler methods—are often disregarded in empirical evaluations, and only neural methods are used as baselines despite their possibly unclear competitiveness.

In some papers, little is also said about hyper-parameter optimization for the baselines. In addition, the code which is used in the optimization and evaluation procedures is not always shared, making reproducibility an issue. With our work, we provide a framework based on the work from [5, 15], where various algorithms can be benchmarked under the exact same conditions, using different evaluation schemes. Overall, we hope that this environment is helpful for other researchers to achieve higher levels of reproducibility and faster progress in this area.

REFERENCES

[1] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. 2009. Improvements That Don't Add Up: Ad-hoc Retrieval Results Since 1998. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. 601–610.

[2] Gabriel de Souza Pereira Moreira, Felipe Ferreira, and Adilson Marques da Cunha. 2018. News Session-Based Recommendations using Deep Neural Networks. In *Proceedings of the 3rd Workshop on Deep Learning for Recommender Systems (DLRS) '18*.

[3] Hailin Fu, Jianguo Li, Jiemin Chen, Yong Tang, and Jia Zhu. 2018. Sequence-Based Recommendation with Bidirectional LSTM Network. In *Advances in Multimedia Information Processing (PCM '18)*. 428–438.

[4] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18)*. 843–852.

[5] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *Proceedings International Conference on Learning Representations (ICLR '16)*.

[6] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y. Chang. 2018. Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. 505–514.

[7] Dietmar Jannach and Malte Ludewig. 2017. When Recurrent Neural Networks Meet the Neighborhood for Session-Based Recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. 306–310.

[8] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17)*. 1419–1428.

[9] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from History and Present: Next-item Recommendation via Discriminatively Exploiting User Behaviors. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. 1734–1743.

[10] Jimmy Lin. 2019. The Neural Hype and Comparisons Against Weak Baselines. *SIGIR Forum* 52, 2 (Jan. 2019), 40–51.

[11] Xiang Lin, Shuzi Niu, Yiqiao Wang, and Yucheng Li. 2018. K-plet Recurrent Neural Networks for Sequential Recommendation. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR '18)*. 1057–1060.

[12] Zachary C. Lipton and Jacob Steinhardt. 2018. Troubling Trends in Machine Learning Scholarship. arXiv:1807.03341 Presented at ICML '18: The Debates.

[13] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '18)*. 1831–1839.

[14] Pablo Loyola, Chen Liu, and Yu Hirate. 2017. Modeling User Session and Intent with an Attention-based Encoder-Decoder Architecture. In *Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys '17)*. 147–151.

[15] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of Session-based Recommendation Algorithms. *User-Modeling and User-Adapted Interaction* 28, 4–5 (2018), 331–390.

[16] Dietmar Jannach Maurizio Ferrari Dacrema, Paolo Cremonesi. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*.

[17] Fei Mi and Boi Faltings. 2018. Context Tree for Adaptive Session-based Recommendation. *CoRR* (2018). arXiv:1806.03733

[18] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. 2002. Using Sequential and Non-Sequential Patterns in Predictive Web Usage Mining Tasks. In *Proceedings International Conference on Data Mining (ICDM '02)*. 669–672.

[19] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *Comput. Surveys* 51 (2018), 1–36. Issue 4.

[20] Jiaxi Tang and Ke Wang. 2018. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM '18)*. 565–573.

[21] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. 2019. A Simple Convolutional Generative Network for Next Item Recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining (WSDM '19)*. 582–590.

User-Centric Evaluation of Session-Based Recommendations for an Automated Radio Station

Malte Ludewig
TU Dortmund, Germany
malte.ludewig@tu-dortmund.de

Dietmar Jannach
University of Klagenfurt, Austria
dietmar.jannach@aau.at

ABSTRACT

The creation of an automated and virtually endless playlist given a start item is a common feature of modern media streaming services. When no past information about the user's preferences is available, the creation of such playlists can be done using session-based recommendation techniques. In this case, the recommendations only depend on the start item and the user's interactions in the current listening session, such as "liking" or skipping an item.

In recent years, various novel session-based techniques were proposed, often based on deep learning. The evaluation of such approaches is in most cases solely based on offline experimentation and abstract accuracy measures. However, such evaluations cannot inform us about the quality as perceived by users. To close this research gap, we have conducted a user study (N=250), where the participants interacted with an automated online radio station. Each treatment group received recommendations that were generated by one of five different algorithms. Our results show that comparably simple techniques led to quality perceptions that are similar or even better than when a complex deep learning mechanism or Spotify's recommendations are used. The simple mechanisms, however, often tend to recommend comparably popular tracks, which can lead to lower discovery effects.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Collaborative filtering**; **Music retrieval**.

KEYWORDS

Session-based Recommendation; Music Recommendation; Quality Perception

ACM Reference Format:
Malte Ludewig and Dietmar Jannach. 2019. User-Centric Evaluation of Session-Based Recommendations for an Automated Radio Station. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3298689.3347046

RecSys '19, September 16–20, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-6243-6/19/09…$15.00
https://doi.org/10.1145/3298689.3347046

1 INTRODUCTION

Modern online media streaming services often provide the functionality of an automated *radio* or *playlist*, where users provide a start item (e.g., a song or a video), and the system then automatically plays a virtually endless list of related items. In some cases, for example, on YouTube or Spotify, users can also give feedback on the played items, e.g., by "liking" them, by explicitly expressing that they are not interested, or by skipping to the next item. In the best case, this feedback is immediately taken into account by the system, leading to an update of the items in the playing queue.

In case the system is used by a new or anonymous user, playlists can be generated with *session-based* recommendation techniques [23]. These approaches do not require long-term preference profiles but are able to make recommendations *solely* based on the most recent interactions of a user. Due to their practical importance also in other domains, e.g., in e-commerce settings, a number of algorithmic proposals for session-based recommendation were made over the years. Technically, different strategies can be applied from traditional association rule mining, over nearest neighbor methods, to recent deep learning based techniques [10, 15, 16].

In the research literature, the comparison of session-based recommendation techniques is mostly based on *offline* experimentation. A recent comparison of techniques in [18], to some surprise, indicated that comparably simple approaches often work at least as good as one of today's more complex state-of-the-art techniques in terms of measures like precision and recall. Given these results, the question arises if such simple techniques only perform well when using such measures or if they are actually able to generate recommendations that are also perceived to be of high quality by users.

To address this question, alongside an offline evaluation, we conducted a between-subjects user study (N=250), where participants interacted with an online radio application that was developed for the experiment. The participants could provide a start track and then received automated recommendations. We compared five algorithms, which included both simple and more complex ones, and the recommendations retrieved through Spotify's API. We analyzed both the participants' observed behavior (e.g., in terms of the number of liked or skipped tracks) as well as the answers they provided in a post-task questionnaire.

One main outcome of the study is that some simple methods are comparable or even better in terms of their quality perception than complex ones, which means simple methods are not only competitive in offline evaluations. However, the tested simpler methods, by design, often recommend more popular tracks, which might lead to fewer opportunities for item discovery. Beyond the specific aspects investigated in the study, our work therefore emphasizes the known need for multi-faceted evaluation approaches and the consideration of domain-specific aspects when comparing algorithms.

2 RELATED WORK

2.1 Next-Track Recommendation Algorithms

A simple technique to determine the next tracks to play is to count track co-occurrences in past sessions and apply a recommendation scheme of the form "Customers who bought … also bought …". Such an approach can be considered to be one of the simplest session-based techniques, which is widely used in practical applications and which, depending on the domain, can already lead to good performance [18]. Over the years, however, a variety of more elaborate next-item prediction techniques for different domains were proposed, from sequential pattern mining over Markov models, embedding-based approaches, and, most recently, to deep learning techniques [4, 8, 9, 15, 16, 21, 25].

In particular in the music domain, also application-specific techniques were applied. Recent research for example showed that recommending popular tracks from artists that are similar to those that the user currently listens to can be quite effective, both in terms of accuracy measures [3] and in terms of the users' quality perception [13]. Similarly, also quite simple nearest-neighbor techniques proved to be effective not only in the music domain, but also in others, e.g., in e-commerce [18]. However, such neighborhood-based methods can exhibit a bias to recommend mostly popular items. The authors of [19] therefore propose to adjust the ranking of items based on their popularity.

In our study, we will compare algorithms of different families. Specifically, we consider one deep learning technique, an artist-based collaborative method, a popularity-aware nearest-neighbor technique, as well as simple item co-occurrences. In addition, we consider the recommendations by a commercial service, in our case the recommendations provided through Spotify's API.

2.2 User Studies

Differently from recent user studies that are focusing on specific aspects of recommendations—such as item similarity [27], presentation aspects [26], the inaction of users [29] or the effects of playlist recommenders on users [12]—our goal is to assess the *quality perception* of different session-based recommendation algorithms in a realistic scenario.

The number of user studies on the perception of music recommendations is comparably low. The two most similar examples of past research are [1] and [13]. In [1], the authors compared a number of "playlisting" approaches through a user study. In their experiment, the participants evaluated entire playlists—two at a time—as produced by different algorithm for a given seed song. They included collaborative techniques, an artist-based one, as well as Apple's iGenius system. In [13], in contrast, the participants did not assess a set of generated playlists, but had to indicate the suitability of four alternative tracks as possible continuations for a given playlist. From a technical viewpoint, the authors considered two neighborhood-based techniques and an artist-based one when determining the possible continuations.

While these two studies share some similarities with our work, their focus and experiment designs were different. Instead of evaluating entire playlists or individual continuations, our goal is to assess the users' quality perceptions in a more interactive setting that is common on modern music streaming sites. Specifically, in our study, the participants can give immediate feedback to the currently played tracks and the system will then automatically update the recommendations accordingly. Similar to [1], we also include a commercial playlisting service in our study. And, like in [13], we contrast offline accuracy results with the users' quality perceptions.

Outside the music domain, various user studies on the quality perception of recommenders were made in the past, e.g., for the movie domain [6, 28]. Recent insights show that it is important that study participants actually "consume" the items they assess [11, 17], which is also the case in our study.

Generally, user studies often rely on user-centric evaluation frameworks as proposed in [14] and [22]. While we do not directly apply these general frameworks in our very specific setup, we partly based our questionnaire items on the considerations presented in these frameworks.

3 STUDY DESIGN

Our main research goal is to understand how different algorithms affect the quality perception of users. Therefore, we developed an own online radio station to conduct the corresponding study.

3.1 Tasks for Participants

(1) Before using the radio, the participants were informed about their tasks and the expected duration of the study. They were also asked to provide informed consent to the terms of the study.

(2) They were then directed to a search interface, where they could enter a query to find a start track. They could listen to excerpts of the retrieved tracks and select one of them to start the radio.

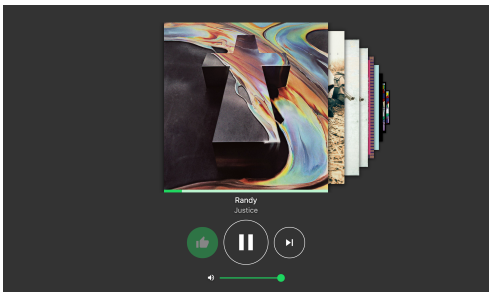


Figure 1: Radio Interface

(3) After this selection, the main radio application was started, see Figure 1. The radio then played the 30-seconds representative track excerpts provided by Spotify. Besides pausing the track and skipping to the next track, the participants could also use a "thumbs up" button to express that they like the current track. At all times, the radio interface provided a visual cue for the users that indicates that there is a list of upcoming tracks. This list was updated after a "thumbs up" or skip action, i.e., the participants received feedback that their actions had an effect.

For each played track, the participants were also asked to provide information if (i) they already knew the track, (ii) if the track matched the previously liked tracks, and (iii) if they liked the track in general, see Figure 2. Proceeding to the next track was only possible after the responses for each track were provided. Overall, each participant was asked to interact with and rate at least 15 tracks.

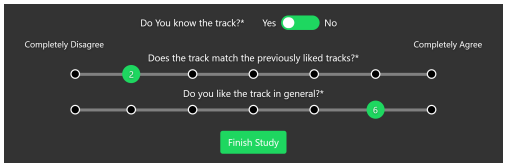


Figure 2: Rating Interface

(4) Once enough tracks were listened to, the participants could proceed to finish the study. In this final part, the participants were asked 11 questions (using 7-point Likert-type items) about their quality perceptions and intention to share or reuse the system. Table 1 shows these 11 questions, where the additional question Q8 is an attention check to assess if the participants answered the questionnaire with care. Furthermore, we asked four questions about their music enthusiasm (not shown in the table).

Table 1: Questions about Users' Quality Perceptions

	Question
Q1	I liked the automatically generated radio station.
Q2	The radio suited my general taste in music.
Q3	The tracks on the radio musically matched the track I selected in the beginning.
Q4	The radio was tailored to my preferences the more positive feedback I gave.
Q5	The radio was diversified in a good way.
Q6	The tracks on the radio surprised me.
Q7	I discovered some unknown tracks that I liked in the process.
Q8	I am participating in this study with care so I change this slider to two.
Q9	I would listen to the same radio station based on that track again.
Q10	I would use this system again, e.g., with a different first song.
Q11	I would recommend this radio station to a friend.
Q12	I would recommend this system to a friend.

3.2 Recommendation Strategies

The independent variable in our study is the assigned recommendation algorithm[1]. We employed five different strategies (see Table 2) and relied on Spotify's Million Playlist Dataset[2], abbreviated as MPD, as a basis for the recommendations.

Playing several tracks of the same artists within a short period of time is uncommon for automated radio stations. Since most tested methods are not designed to take this domain particularity into account, we designed a diversifying post-processing strategy, which we applied to all playlists returned by the algorithms. Specifically, we made sure that there are no artist repetitions within the next 3 tracks by re-ranking the tracks in the playlist. For the recommendations returned by Spotify's API, we furthermore removed tracks that did not appear in the MPD dataset. Thereby, we guaranteed a fair comparison of the techniques, in which all approaches recommend from the same catalog of items.

We optimized the hyper-parameters for the different strategies in an offline experiment on a subset of the MPD dataset. From a random sub-sample of 100,000 playlists we randomly selected 2,000 playlists as a validation set, while using the remaining playlists for model training. Parameter tuning was performed with a randomized search strategy in 100 iterations to find the best configuration for the mean reciprocal rank at list length 5.

[1]We used a round-robin assignment scheme as done in [13].
[2]https://recsys-challenge.spotify.com/

Table 2: Tested Recommendation Strategies

AR	A simple method based on association rules of length two, see [18].
CAGH	Recommends the greatest hits of artists similar to those liked in the current session [3]. The similarity is based on artist co-occurrences in user-provided playlists and approximated with matrix factorization.
GRU4REC	A recent session-based algorithm based on Recurrent Neural Networks (v2.0) [9]. The algorithm hyper-parameters were optimized for the mean reciprocal rank at list length 5.
S-KNN	A session-based nearest-neighbor approach proposed in [19] that lowers the predicted relevance scores for highly popular items. We used 500 as the number of neighbors, and set the sample size to 1000.
SPOTIFY	Recommendations in this treatment group were retrieved in real time from Spotify's API. Tracks that are not present in the MPD were excluded from the recommendations.

4 RESULTS

We have recruited 316 participants over Amazon's Mechanical Turk crowdsourcing platform ("Masters" only), ending up with reliable submissions from 250 unique users, i.e., 50 in each treatment group. The remaining submissions were eliminated as the users did not pass the attention check. On average, the participants needed about 15 minutes to complete the task. A majority of the users (nearly 80%) was from the US; over 50% of the participants were aged between 25 and 34.

On average and across all treatment groups, the participants listened to around 16 tracks (slightly above the minimum requirement), with an average pure listening time of 5.5 minutes. There were no significant differences in these respects.

Number of likes. The average number of *likes* per user were as follows: SPOTIFY: 4.48, GRU4REC: 5.36, CAGH: 5.38, S-KNN: 5.63, AR: 6.48. The AR method led to significantly[3] more likes than CAGH, SPOTIFY, and GRU4REC ($p<0.05$). Furthermore, the S-KNN method received 5.6 likes on average, which was significantly higher than SPOTIFY as well. All other differences were not significant.

Popularity. Looking at the average popularity of the recommended tracks[4], we found that SPOTIFY's and GRU4REC's recommendations were the least popular ones while AR and CAGH tend to recommend mostly popular items. Generally, the number of received likes per playlist highly correlates with the average track popularity of a list (r=0.89).

Individual Track Ratings. Table 3 shows our observations regarding the feedback for the individual tracks as shown in Figure 2. In terms of the percentage of already known tracks, we see that Spotify's recommendations are significantly less often known (or: more novel) than those of the other techniques.

When asking participants to what extent each track matches the previously liked ones, we observe that S-KNN and CAGH work best, while AR leads to recommendations that match the current playlist the least. The differences between S-KNN and the methods AR, GRU4REC, and SPOTIFY are significant ($p<0.05$).

Finally, looking at the average track "rating" in general, the differences between the algorithms are small. Somewhat surprisingly, the tracks produced by the AR method were the least liked ones, often with a significant difference, even though the average track

[3]Throughout the work, we use one-way ANOVA and a subsequent Tukey post-hoc test when the pre-requisites for these tests are fulfilled. Otherwise, we applied a Kruskal-Wallis test and a subsequent Mann-Whitney-U test.
[4]Computed based on the number of track occurrences in the MPD dataset.

popularity was the highest and they received the most like statements. This phenomenon can be explained when looking at the *mode* of the answers. The most frequent response (mode) was 1 for AR, whereas it was 7 for all other recommendation strategies. This means that AR probably recommended many "controversial" tracks that the users did not like even though they are very popular, e.g., because they matched previous tracks the least.

Table 3: Statistics for Item-Specific Questions (Mean and Standard Deviations)

Algorithm	Track known (%)	Track matches the playlist	Like the track in general
AR	8.61	4.06 ±1.60	4.34 ±1.37
CAGH	10.83	5.15 ±1.14	5.03 ±1.22
GRU4REC	9.30	4.61 ±1.52	4.94 ±1.31
S-KNN	10.13	5.31 ±1.04	4.94 ±1.06
SPOTIFY	7.00	4.72 ±1.13	4.69 ±1.07

Post-Task Questionnaire. For the Questions Q1 to Q12, we looked mostly at the median and mode values[5] and analyzed differences with the non-parametric tests.

For **Q1**, we found that the recommendations of S-KNN (median: 6)[6] were significantly more liked than those of AR, GRU4REC and SPOTIFY ($p<0.05$) and were perceived to be slightly better also than CAGH. All methods matched the user's general taste well (**Q2**), with median values of 5 (AR, SPOTIFY) or 6 (other methods). The values for AR were significantly lower than for CAGH, S-KNN, and GRU4REC.

The general pattern that AR performs worst and S-KNN best was also found for **Q3**, where S-KNN performed significantly better at finding tracks that match the seed track than AR and also GRU4REC. Regarding the adaptiveness of the radio upon user feedback (**Q4**), the median values were generally comparably high, ranging between 5 and 6. The analysis revealed that only AR performed significantly worse than CAGH, S-KNN, and SPOTIFY.

No differences between the group were found however regarding the diversification of the radio (**Q5**) and the surprise level (**Q6**). In terms of the discovery of unknown but liked tracks (**Q7**), SPOTIFY excelled. The responses were higher than for CAGH, S-KNN, GRU4REC, and AR ($p<0.1$). For the other methods, no significant differences could be found.

Regarding the last block of questions (**Q9** to **Q12**) about the users' intention to reuse the system or recommend it to friends, the values for S-KNN, CAGH and SPOTIFY are slightly higher than for the other techniques. The differences between these three methods and AR were always significant ($p<0.05$). For **Q10** and **Q12**, the responses for S-KNN and CAGH were also significantly higher than for GRU4REC.

4.1 Offline Accuracy vs. User Experience

The final goal of our work was to compare the users' quality perception with accuracy results obtained from offline experiments, since previous work [2, 5, 7, 20, 24] suggests that there can be a discrepancy. We therefore evaluated the different algorithms using five random sub-samples of the MPD dataset following the same procedure as for the parameter optimization described in Section 3.2. Furthermore, we applied the sequential evaluation protocol from [9]. In our work, we measure *precision* (P) and *recall* (R) in the usual way, comparing the list of recommended items with the next tracks in the playlist. In addition, we report the specific measurement method used in [9], which only considers the immediate next item, using the *hit rate* (HR) and the *mean reciprocal rank* (MRR). In order to design the evaluation as close as possible to the online application, we only proceeded through the first 15 entries of each test playlist to match the number of necessary interactions in the study.

Table 4: Offline Results

Algorithm	P@5	R@5	HR@5	MRR@5
S-KNN	**0.271**	**0.044**	0.137	0.077
GRU4REC	0.161	0.028	**0.151**	**0.096**
AR	0.234	0.037	0.135	0.081
CAGH	0.172	0.024	0.052	0.026
SPOTIFY	0.009	0.001	0.002	0.001

Table 4 shows that the ranking of the algorithms depends on the choice of the measurement method. The results indicate that the usual precision and recall measurement correlates better with our observations of the users' perception, with the S-KNN method leading to the best results in the offline experiment. The performance of SPOTIFY is very low in this comparison. In some ways, this is in line with the observations from the user study, which indicates that SPOTIFY's algorithm seems to be more optimized for discovery than for precision or recall on historical data.

5 SUMMARY OF FINDINGS

We have conducted a user study that investigates the quality perception of adaptive music recommendations in an automated radio station and have contrasted the findings with an offline experiment. To our knowledge, this is the first study of this type in the context of session-based recommendation techniques.

Bearing in the mind that the number of participants might be a possible limitation of this work, the main findings of the study are as follows. First, using a comparably simple nearest-neighbor technique led to radio stations that were favorable, in more than one dimension, over a station that was built on a more complex deep learning technique. Second, the AR method, which recommended the most popular tracks, led to the highest number of likes, but was performing poorly in most other dimensions; optimizing and evaluating algorithms based on explicit like statements can therefore be misleading. Third, SPOTIFY's algorithm was better than all other methods in helping users discover new tracks they like. At the end, no differences in the participants' intention to re-use or recommend the system were found. The ability of SPOTIFY's algorithm to support discovery might therefore compensate other aspects where the algorithm did not excel. Fourth, the offline experiments indicate that precision and recall can be indicative of the quality of the recommendations to some extent. In general, however, the good quality perception of SPOTIFY's algorithm despite the very low offline accuracy results emphasizes that factors other than accuracy can be decisive for the long-term adoption of a system.

[5] Using means for *single* Likert-scale type items is considered potentially unreliable even though we only used labels for the ends of the numerical scale.
[6] A full table of the results is provided as auxiliary material in the ACM Digital Library.

REFERENCES

[1] Luke Barrington, Reid Oda, and Gert R. G. Lanckriet. 2009. Smarter than Genius? Human Evaluation of Music Recommender Systems. In *Proc. ISMIR '09*. 357–362.

[2] Jöran Beel and Stefan Langer. 2015. A Comparison of Offline Evaluations, Online Evaluations, and User Studies in the Context of Research-Paper Recommender Systems. In *Proc. TPDL '15*.

[3] Geoffray Bonnin and Dietmar Jannach. 2014. Automated Generation of Music Playlists: Survey and Experiments. *Comput. Surveys* 47, 2 (2014), 26:1–26:35.

[4] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist Prediction via Metric Embedding. In *Proc. KDD '12*. 714–722.

[5] Paolo Cremonesi, Franca Garzotto, and Roberto Turrin. 2012. Investigating the Persuasion Potential of Recommender Systems from a Quality Perspective: An Empirical Study. *Transactions on Interactive Intelligent Systems* 2, 2, Article 11 (June 2012), 11:1–11:41 pages.

[6] Michael D. Ekstrand, F. Maxwell Harper, Martijn C. Willemsen, and Joseph A. Konstan. 2014. User Perception of Differences in Recommender Algorithms. In *Proc. RecSys '14*. 161–168.

[7] Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, and Amr Huber. 2014. Offline and Online Evaluation of News Recommender Systems at Swissinfo.ch. In *Proc. RecSys '14*.

[8] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2014. Context Adaptation in Interactive Recommender Systems. In *Proc. RecSys '14*. 41–48. http://doi.acm.org/10.1145/2645710.2645753

[9] Balázs Hidasi and Alexandros Karatzoglou. 2017. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. *CoRR* abs/1706.03847 (2017).

[10] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based Recommendations with Recurrent Neural Networks. In *Proc. ICLR '16*.

[11] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. 2015. Item Familiarity as a Possible Confounding Factor in User-Centric Recommender Systems Evaluation. *i-com Journal of Interactive Media* 14, 1 (2015), 29–39.

[12] Iman Kamehkhosh, Geoffray Bonnin, and Dietmar Jannach. 2019. Effects of Recommendations on the Playlist Creation Behavior of Users. *User Modeling and User-Adapted Interaction* May (2019).

[13] Iman Kamehkhosh and Dietmar Jannach. 2017. User Perception of Next-Track Music Recommendations. In *Proc. UMAP '17*.

[14] Bart P. Knijnenburg, Martijn C. Willemsen, Zeno Gantner, Hakan Soncu, and Chris Newell. 2012. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction* 22, 4 (2012), 441–504.

[15] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural Attentive Session-based Recommendation. In *Proc. CIKM '17*. 1419–1428.

[16] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation. In *Proc. KDD '18*. 1831–1839.

[17] Benedikt Loepp, Tim Donkers, Timm Kleemann, and Jürgen Ziegler. 2018. Impact of Item Consumption on Assessment of Recommendations in User Studies. In *Proc. RecSys '18*. 49–53.

[18] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of Session-based Recommendation Algorithms. *User-Modeling and User-Adapted Interaction* 28, 4–5 (2018), 331–390.

[19] Malte Ludewig, Iman Kamehkhosh, Nick Landia, and Dietmar Jannach. 2018. Effective Nearest-Neighbor Music Recommendations. In *Proc. RecSys '18 Challenge Workshop at ACM RecSys*.

[20] Andrii Maksai, Florent Garcin, and Boi Faltings. 2015. Predicting Online Performance of News Recommender Systems Through Richer Evaluation Metrics. In *Proc. RecSys '15*.

[21] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. 2002. Using Sequential and Non-Sequential Patterns in Predictive Web Usage Mining Tasks. In *Proc. ICDM '02*. 669–672.

[22] Pearl Pu, Li Chen, and Rong Hu. 2011. A User-centric Evaluation Framework for Recommender Systems. In *Proc. RecSys '11*. 157–164.

[23] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *Comput. Surveys* 51 (2018), 1–36. Issue 4.

[24] Marco Rossetti, Fabio Stella, and Markus Zanker. 2016. Contrasting Offline and Online Results when Evaluating Recommendation Algorithms. In *Proc. RecSys '16*.

[25] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-Based Recommender System. *Journal of Machine Learning Researh* 6 (2005), 1265–1295.

[26] Longqi Yang, Michael Sobolev, Christina Tsangouri, and Deborah Estrin. 2018. Understanding User Interactions with Podcast Recommendations Delivered via Voice. In *Proc. RecSys '18*. 190–194.

[27] Yuan Yao and F. Maxwell Harper. 2018. Judging Similarity: A User-centric Study of Related Item Recommendations. In *Proc. RecSys '18*. 288–296.

[28] Yuan Yao and F. Maxwell Harper. 2018. Judging Similarity: A User-centric Study of Related Item Recommendations. In *Proc. RecSys '18*. 288–296.

[29] Qian Zhao, Martijn C. Willemsen, Gediminas Adomavicius, F. Maxwell Harper, and Joseph A. Konstan. 2018. Interpreting User Inaction in Recommender Systems. In *Proc. RecSys '18*. 40–48.