Multicriteria Evaluation for Top-k and Sequence-based Recommender Systems


Doctoral Dissertation
Doctoral Program in Computer and Control Engineering (32^{th} cycle)

Multicriteria Evaluation for Top-k and Sequence-based Recommender Systems

Diego Michele Monti
* * * * * *

Supervisors:
Prof. Maurizio Morisio, Supervisor
Dr. Giuseppe Rizzo, Co-supervisor

Doctoral Examination Committee:
Prof. Luis Martínez-López, Referee, Universidad de Jaén
Prof. Olga C. Santos, Referee, Universidad Nacional de Educación a Distancia
Prof. Andrea Bottino, Politecnico di Torino
Prof. Marco Torchiano, Politecnico di Torino
Dr. Iacopo Vagliano, Amsterdam UMC

Politecnico di Torino
May 7^{th}, 2020

I hereby declare that, the contents and organisation of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Diego Michele Monti
Turin, May 7th, 2020

Summary

Context The need of effectively identifying relevant items from a potentially overwhelming catalog has led to the creation of automatic approaches for supporting users of online platforms in such a time-consuming task. Recommender systems are software tools and algorithms designed to suggest items to users according to their preferences. The traditional goal of the recommendation problem is to create a ranked list of suggestions for each user. However, a more novel paradigm is represented by algorithms capable of exploiting the temporal dimension of the available ratings, known as sequence-based recommender systems. Differently from the field of information retrieval, performing an offline experiment for comparing multiple recommendation approaches is a challenging task, as the ground truth is represented by the subjective preferences of the users collected before the introduction of the system under evaluation. Nevertheless, it represents a powerful tool for selecting the most promising approaches to be further tested in subsequent trails.

Goal In this dissertation, we study how to perform the offline evaluation of a generic recommender system by exploiting a multicriteria approach that relies on a set of heterogeneous metrics. We provide an answer to the following research questions: what is the current state-of-the-art regarding multicriteria recommender systems and how their are evaluated in literature; how can a multicriteria evaluation approach be exploited for comparing different sequence-based recommender systems; what is the most suitable protocol for performing an offline evaluation of a top-k recommender system; and to what extent the structure of a rating dataset can influence the results of an offline evaluation.

Method To answer the first question, we investigate the topic of multicriteria recommender systems with a systematic literature review. Regarding the second question, we propose an evaluation framework called Sequeval designed to compare in a replicable way different sequence-based recommenders considering eight dimensions. With respect to the third question, we introduce RecLab, an evaluation toolkit designed following a distributed approach to overcome some limitations of the currently available evaluation solutions. Finally, for addressing the fourth question, we describe a method based on data visualization to explore the structure

of a rating dataset and we report on an algorithm capable of generating alternative versions of an existing collection of ratings.

Results From the systematic literature review, we observed that is not possible to directly compare the results obtained with different multicriteria recommendation approaches due to the extreme variability in the reported experimental protocols. We exploited Sequeval and RecLab to successfully conduct some experimental campaigns involving different recommenders and datasets. The availability of multicriteria metrics enables the experimenter to obtain a more comprehensive picture of the systems under consideration. We validated our visualization method and generative approach by qualitatively and quantitatively comparing the results obtained with different rating datasets. Finally, we considered two novel recommendation approaches as possible use cases of multicriteria evaluation methods.

Conclusion In summary, this dissertation deals with the problem of conducting an offline comparison of recommender systems considering both traditional and sequential scenarios. We designed the proposed frameworks for addressing the most critical problems that emerged from literature, namely the non-reproducibility of the results, the comparability of different studies, and the bias of relying on a few metrics only. The software code of our frameworks is freely available on the Web in an attempt to foster further reuse and extension.

Acknowledgements

This dissertation represents the end of a journey that lasted about three years and a half and it would not have been possible without the help and the continuous support of different people with whom I had the privilege to collaborate.

First of all, I would like to thank my supervisor, prof. Maurizio Morisio, for encouraging me to pursue a doctorate title and for always helping to face the difficulties that such a choice necessarily implies, and my co-supervisor, Giuseppe Rizzo, for always trying, and hopefully succeeding, in making me a more mature researcher. Indeed, I would like to thank prof. Marco Torchiano, for recruiting me as a teaching assistant since my first year and for introducing me to the topic of data visualization, and Luca Ardito, for the always precious advice and suggestions.

A very special thank goes to Iacopo Vagliano, for helping to orientate myself in the complex world of recommender systems during my early days, and to Enrico Palumbo, for the countless hours spent together debating mathematical equations and configuration parameters of our experiments.

Of course, I would like to mention all the colleagues and friends who contributed to the positive and vibrant atmosphere of Lab 1, strictly in alphabetical order: Alysson Dos Santos, Amirhosein Toosi, Edoardo Battegazzorre, Erion Çano, Francesco Strada, Isabeau Oliveri, Mariachiara Mecati, Riccardo Coppola, Rifat Rashid, and Simone Leonardi.

Several experimental activities that I conducted would not have been possible without the computational resources kindly provided by the HPC@POLITO project (http://hpc.polito.it), to which I am grateful.

Last but not least, I would like to thank my parents for their unconditional love and fundamental support during this journey.

Contents

List of Tables

List of Figures

Glossary

collaborative filtering (CF): a recommendation approach based on the rating behaviour of similar users on similar items.

content-based recommender system: a recommendation approach based on the preferences of the users and the characteristics of the items.

evaluation framework: an evaluation protocol, a set of evaluation metrics, and, if available, a software implementation of them.

evaluation protocol: the procedure that needs to be followed by a researcher to perform an experiment.

hybrid recommender system: a recommendation approach that combines different techniques to select the suggested items.

knowledge-based recommender system: a recommendation approach based on externally encoded domain knowledge.

multicriteria evaluation: an experimental approach based on a comprehensive set of different evaluation metrics.

multicriteria recommender system: a recommender system that exploits multiple ratings per item expressed over different criteria.

offline evaluation: an experimental approach based on user preferences collected before the introduction of the system under evaluation.

online evaluation: an experimental approach that consists in making a novel recommender system available to a potentially large community of users.

rating dataset: a dataset containing the preferences expressed by a set of users over a catalog of items at a certain timestamp.

recommender system (RS): a software tool and an algorithm designed to suggest items to users according to their preferences.

sequence-based recommender system: a recommender system capable of suggesting personalized sequences of items.

sequential recommender system: a recommender system designed to consider the timestamp associated with user ratings.

synthetic dataset: a rating dataset created in an artificial way.

top-k recommender system: a recommender system capable of suggesting a list of the k most relevant items per user.

user study: an experimental approach that consists in making a recommender system available to a limited set of subjects in a controlled environment.

Chapter 1

Introduction

Due to the large variety of products and digital content available on the Web, an increasing number of people are interested in obtaining personalized suggestions, in order to reduce the effort of inspecting all the items in a catalog for selecting the best one according to their preferences. An automated tool capable of recommending items to users in a personalized way is defined as a *recommender system* [112].

Recommender systems (RS) were initially conceived at the beginning of the 1990s [51] and, nowadays, they are considered part of a research field that is independent from information retrieval [61]. In fact, while search engines are based on queries and, therefore, they react to user stimuli, recommender systems try to automatically identify items that could be of interest for a certain user [11].

A popular recommendation technique, called collaborative filtering (CF), consists of learning users' preferences by only relying on their interactions with the items available in a catalog. For example, using a nearest-neighbor search or a machine learning model, it is possible to select the most relevant items for each user who has interacted enough with the system [122]. An alternative approach to this problem is represented by content-based recommenders, which can generate suggestions by matching users' profiles with the features of the items [11, 12]. Another family of recommendation methods proposed in the literature is represented by hybrid algorithms that are capable of combining both collaborative and content-based filtering for mitigating the individual weaknesses of the previous techniques [120].

In a collaborative filtering setting, users are typically required to rate items that are already familiar with, relying on a numerical scale, for example a 5-star scale. This value should objectively represent the utility that the user gained from the consumption of that item. Given a sufficient number of users and ratings, a recommender system is capable of predicting the utility score that a user would assign to unrated items [4]. The items with the highest predicted rating are finally suggested to the user, in a top-k ranked list.

A well-established line of research is related to the minimization of the error in predicting these values. However, such a task has limited practical applications,

because several techniques capable of predicting ratings with a high accuracy are already available [102]. Furthermore, accurate suggestions may sometimes not be the most useful ones for users [61]. For example, predicting high ratings for a popular item is probably accurate, but not meaningful, as users are likely to be already aware of that item.

In recent years, this traditional approach has been put aside in favor of others closer to the needs of the users. For example, many recommender systems now rely on binary or implicit signals in order to create a personalized experience [55]. Those signals are more intuitive to be understood and easier to be generated. Another important factor that has started to be considered as a possible input of the recommender is the temporal dimension of the preferences [108].

In general, the offline evaluation of recommender systems is a challenging task, because, differently from the field of information retrieval, the ground truth is always uncertain, as it is based on the subjective preferences of the users collected before the introduction of the system under evaluation [62]. In fact, it is widely known that novel recommendation approaches should be evaluated in the context of online experiments involving human subjects in order to obtain reasonably robust results about their performance.

Nevertheless, most of the studies available in literature support their conclusions with offline trails relying on the preferences of users collected without considering the algorithms under investigation [55]. Despite the possible weaknesses of this approach [116], offline experiments are extremely popular among researchers because of their limited costs and the theoretical reproducibility of their results. In industry, they are usually considered a powerful tool for pruning the number of possible recommender systems that need to be tested with real users, thus mitigating the economical impact of eventual failures.

This dissertation discusses how to perform the offline evaluation of a generic recommender system by exploiting a multicriteria approach that relies on a comprehensive set of different metrics. By adopting the proposed protocols, it is possible to obtain a more general picture of the systems under evaluation, avoiding frequent problems like the popularity bias and the non reproducibility of the results. We experiment with multiple offline evaluation techniques in the context of both traditional and sequential recommenders.

Furthermore, we investigate the main characteristics of different rating datasets and how they can influence the performance of the systems that rely on them. Finally, we also consider the topic of multicriteria from a different angle, by analyzing existing methods for exploiting a multifaceted knowledge of users' preferences.

More formally, we answer to the following top-level research questions.

RQ1 What is the current state-of-the-art regarding multicriteria recommender systems and how their are evaluated in literature?

RQ2 How can a multicriteria evaluation approach be exploited for comparing different sequence-based recommender systems?

RQ3 What is the most suitable protocol for performing an offline evaluation of a top-k recommender system?

RQ4 To what extent the structure of a rating dataset can influence the results of an offline evaluation?

As regards RQ1, we conduct a systematic literature review to investigate in dept the field of multicriteria recommender systems, considering the exploited recommendation approaches and how the proposed algorithms were evaluated.

We analyze the different machine learning and data mining techniques typically exploited in literature and we classify them according to the recommendation phase. Furthermore, we review how the proposed algorithms have been evaluated with respect to the experimental settings, the metrics, and the datasets. In Chapter 3, we provide detailed answers to the following research questions.

RQ1.1 What are the most relevant studies addressing multicriteria RSs?

RQ1.2 What are the most challenging problems faced by researchers?

RQ1.3 What are the approaches used by multicriteria recommenders?

RQ1.4 Which techniques and methods have been proposed?

RQ1.5 In which domains multicriteria recommender systems are applied?

RQ1.6 Which protocols and frameworks are used for their evaluation?

RQ1.7 Which metrics are considered during their evaluation?

RQ1.8 Which datasets are used for testing the algorithms?

RQ1.9 What are the most promising directions for future works?

With respect to RQ2, we research and prototype an offline evaluation framework called Sequeval that is designed to evaluate recommender systems capable of suggesting sequences of items, instead of lists of items. In Chapter 4, we provide a set of mathematical definitions to characterize in a precise way what is a recommender system capable of suggesting sequences. In detail, we expand the traditional concept of rating by adding to it the notion of temporal dimension.

Then, we propose to consider a sequence as a temporally ordered list of ratings and a sequence-based recommender as a function that is able to return a sequence given its required length and a seed rating. These definitions are conceived as an

extension of the seminal works on recommenders capable of suggesting sequences available in literature [108].

Starting from this formalization, we propose an evaluation protocol that can be applied to any sequence-based recommender system. First, an initial dataset is transformed into a set of sequences. Then, the available sequences are split between training and test sets. At this point, one or more external recommenders are plugged into the framework: they are exposed to the training sequences and they are asked to create suggested sequences starting from the same seeds of the test ones. Finally, considering the recommendations available, the framework can compute eight different evaluation metrics.

We report the lessons learned using this framework for assessing the performance of four baselines and two recommender systems based on conditional random fields and recurrent neural networks, considering two rating datasets. Sequeval is publicly available and it can be exploited by researchers and practitioners when experimenting with sequence-based recommender systems, providing comparable and objective evaluation results. In Chapter 4, we consider the following research questions.

RQ2.1 What is the formal definition of a sequence-based recommender system?

RQ2.2 How already established metrics can be extended and adapted for evaluating a sequence-based recommender system?

RQ2.3 Against which baseline approaches a sequence-based recommender system can be compared?

For addressing RQ3, we introduce RecLab, an evaluation toolkit discussed in Chapter 5 and based on RESTful APIs that can be used to overcome the problem of evaluating traditional recommenders in heterogeneous settings. In fact, because the recommenders are deployed on different servers, the evaluator does not need to know their implementation details.

The researcher can specify the experimental parameters in a Web-based interface for starting a new evaluation campaign. RecLab will then contact all the recommenders selected as part of the comparison and it will display the results computed considering a comprehensive set of seven different metrics.

We propose a Web-based interaction protocol in order to standardize the procedure for evaluating the recommenders. The evaluator first requests the training of a new model, then it provides the training set created according to the settings of the experiment. When the model is ready, the evaluator asks the recommender to create a list of k suggestions for each user of the test set. The results of all experiments are permanently stored and publicly available in order to support accountability and comparative analyses. In Chapter 5, we provide an answer to the following research questions.

RQ3.1 How can different top-k recommender systems be fairly compared in heterogeneous settings without necessary exposing their algorithms?

RQ3.2 To what extent it is possible to support the reproducibility of the experiments and the accountability of the results?

RQ3.3 How can the availability of different metrics support the experimenter in the interpretation of the obtained results?

Regarding RQ4, we explore a method for visualizing the structure of a rating dataset in Chapter 6 and we discuss how to generate a synthetic dataset for evaluation purposes in Chapter 7.

We introduce a qualitative approach based on data visualization for creating a graphical summary of any collection of user preferences. This method is useful for visually identifying similarities and differences among various rating datasets. In fact, if two datasets result in similar visualizations, the behavior of different recommender systems relying on them will be consistent. Furthermore, we develop a Web-based tool, named RS-viz, for easily constructing the proposed visualization and comparing rating datasets in an intuitive way. In Chapter 6, we consider the following research questions.

RQ4.1 How can data visualization techniques be exploited to create a graphical summary of the main characteristics of a rating dataset?

RQ4.2 To what extent the graphical representation of different rating datasets can be useful to easily identify their similarities and diversities?

Another relevant problem that we try to address in this dissertation is related to the shortage of publicly available rating datasets. In fact, it is necessary to rely on a collection of user preferences obtained in a particular domain to perform an offline experiment, but the availability of such datasets is often limited. Some researchers have started to rely on synthetic ratings. However, the results obtained from these experiments may be questionable, as the generated datasets are usually not capable of capturing the characteristics of a particular domain of interest.

For this reason, we propose an approach for automatically generating synthetic datasets with a configurable number of users leveraging on a reference dataset that is used as the seed of the process and that encodes the peculiarities of a domain of interest. Such a method could also be exploited for anonymizing existing datasets before their public release in the context of privacy-aware suggestions. In Chapter 7, we analyze the following research questions.

RQ4.3 What is the impact of using a synthetic dataset instead of a real one on the results of an offline experiment in the context of recommender systems?

RQ4.4 Can a generative approach be exploited to create a synthetic dataset that exhibits properties similar enough to the ones of a real dataset?

RQ4.5 To what extent this method can be consistently applied to datasets from different domains and of different sizes?

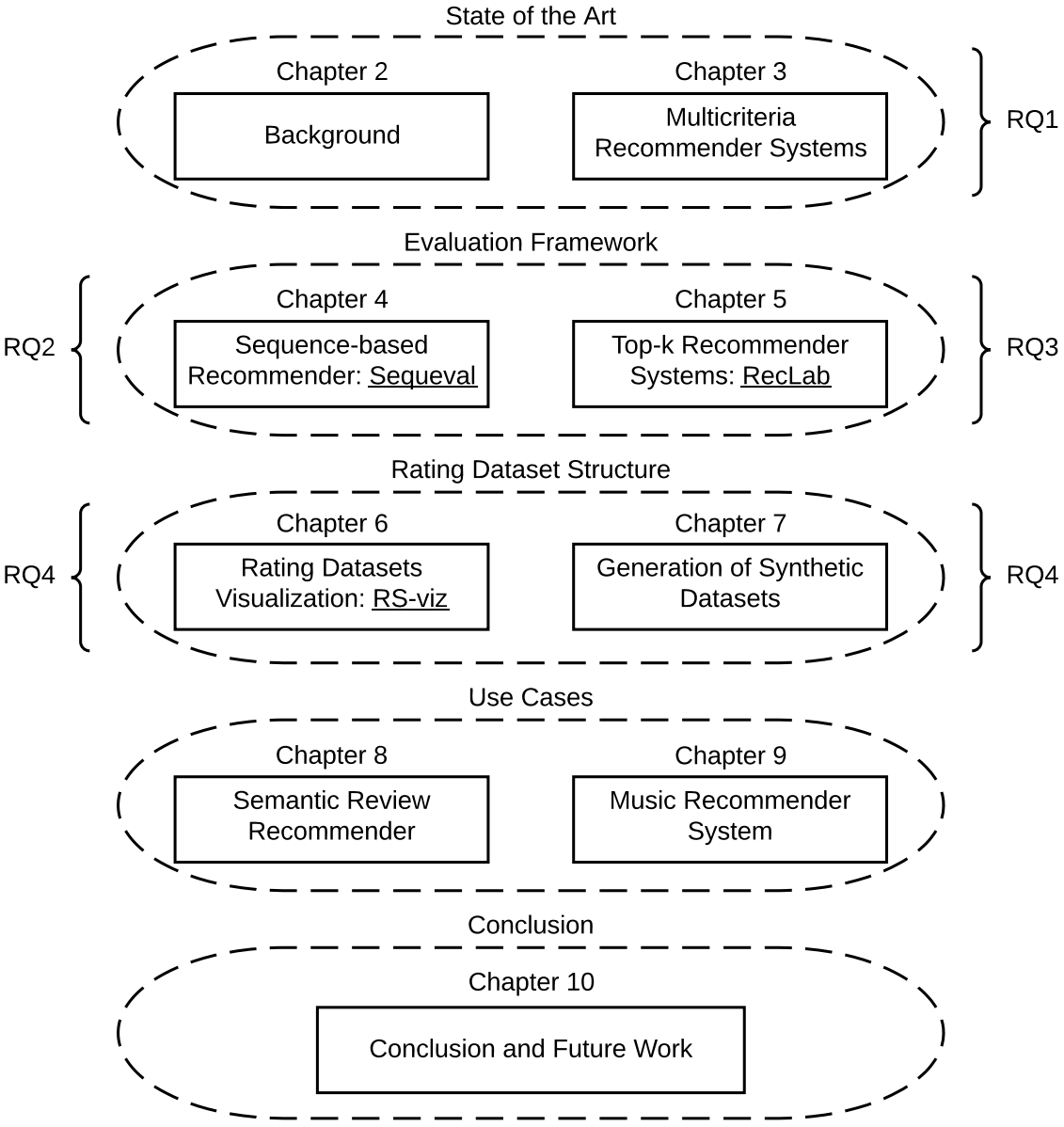


Figure 1.1: An overview of the dissertation structure illustrating its division in chapters, the top-level research questions and the main outcomes of the related research activity. The software tools developed are underlined.

In summary, as depicted in Figure 1.1, this dissertation deals with the problem of conducting an offline comparison of recommender systems from different perspectives. We analyze existing approaches for exploiting multicriteria ratings, with a special emphasis on the methods currently employed for validating the quality of the suggestions. Then, we discuss multicriteria evaluation techniques, considering both sequence-based recommenders as well as more traditional approaches. Furthermore, we explore the problem of selecting the right dataset according to the evaluation context by visualizing its structure and generating alternative versions of it. Finally, we apply our knowledge of multicriteria evaluation techniques to different use cases. We propose and evaluate with an ensemble of metrics a semantic review recommender capable of annotating product reviews to extract useful information from them and a music recommender system designed to create playlists starting from a few songs that represent the seeds of the process.

The remainder of this dissertation is organized as follows. In Chapter 2, we review existing recommendation and evaluation approaches available in literature and the associated challenges. In Chapter 3, we report the results of a systematic literature review dealing with the topic of multicriteria recommender systems. In Chapter 4, we introduce Sequeval, our framework for evaluating sequence-based recommender systems, while, in Chapter 5, we discuss RecLab, a Web-based evaluation toolkit for top-k recommenders. In Chapter 6, we describe a qualitative method for visualizing the ratings available in any collection of users' preferences. In Chapter 7, we present our approach for generating a synthetic dataset that exhibits the same properties of a real one. In Chapter 8, we report on the creation and evaluation of a semantic review recommender, while, in Chapter 9, we describe our approach for automatically generating music playlists. Finally, in Chapter 10, we formulate our conclusions and we discuss open issues and possible future works.

Chapter 2

Background

Nowadays, the amount of information available on the Web is overwhelming. For this reason, the availability of tools capable of selecting from a huge catalog a short list of items that are of potential interest for a particular user is a critical success factor for almost any online platform. Therefore, recommender systems represent one of the technical solutions commonly employed in industry to address the issue of effectively exploring a vast horizon of possible choices.

A related and equally popular approach is represented by search engines. Despite the common roots of these solutions, the task of evaluating with an offline experiment a recommender system is usually more challenging, as the ground truth is represented by subjective, and sometimes even emotional, preferences.

In this chapter, we first review some of the recommendation approaches available in literature (Section 2.1), then we discuss the main challenges associated with their offline evaluation (Section 2.2).

2.1 Recommender Systems

According to Ricci et al. [112], recommender systems are software tools and algorithms designed to suggest items to users according to their preferences.

In general, recommender systems can be classified in different categories based on the recommendation approach. The most widespread categories of recommender systems are content-based, collaborative filtering, knowledge-based, and hybrid [4], as summarized in Table 2.1. Content-based recommenders only rely on the past preferences of the current user in order to construct her profile and select suggested items. In contrast, collaborative filtering approaches analyze the behaviour of similar users for identifying candidate items. A knowledge-based recommender embeds domain-specific knowledge that is used for matching user requirements with items of potential interest. Finally, hybrid approaches combine in many different ways the previous methods. Other less common categories of recommender

systems include community-based and demographic techniques [19]. A community-based recommender also considers the relationships of trust among its users, while a demographic recommender mainly relies on demographic profiles.

Method	Advantages	Disadvantages
Content-based	Transparency, limited privacy issues	Requires a description of the items
Collaborative filtering	Scalability, only based on ratings	Data sparsity and cold start problems
Knowledge-based	No interaction history required	Difficult to the create the model
Hybrid	Combines multiple approaches together	Limited performance

Table 2.1: Comparison of the advantages and disadvantages of the recommendation techniques mentioned in [4].

A detailed analysis of the recommendation approaches available in literature is beyond the scope of this dissertation. Many authors already conducted different studies dealing with the topic of recommender systems. Park et al. [102] reviewed hundreds of journal articles for analyzing the main application fields and data mining techniques exploited by different recommenders. Hong et al. [66] discussed the literature about context-aware recommenders, while Figueroa et al. [44] and Çano et al. [21] conducted systematic literature reviews about Linked Data and hybrid recommender systems respectively.

More recently, Quadrana et al. [108] classified different approaches according to their capability of managing sequences of items. Portugal et al. [105] reviewed commonly exploited machine learning techniques, while Zhang et al. [139] discussed the most promising deep learning methods for generating personalized items.

In the following, we briefly introduce a formal definition of the recommendation problem. Then, we review the recommendation approaches considered in this dissertation, that is top-k, sequential, review-based, and Linked Data-based ones. Please note that sequence-based recommender systems are a generalization of sequential recommenders and they will be defined later in Section 4.1.

2.1.1 Top-k Recommender Systems

Traditionally, recommender systems try to identify for each user an item that maximises the utility that she should gain from the consumption of that item [18]. In other words, recommender systems estimate a utility function $R(\upsilon, \iota)$ that, given a user $\upsilon \in \mathcal{U}$ and an item $\iota \in \mathcal{I}$, predicts if ι should be recommended to υ. In principle, the utility function could be an arbitrary function, including a profit

function. The formal definition of this problem, as provided by Adomavicius & Tuzhilin [2], is reported in Equation 2.1.

$$\forall u \in \mathcal{U}, \quad \iota'_u = \arg\max_{\iota \in \mathcal{I}} R(u, \iota) \tag{2.1}$$

In practice, users are commonly asked to quantify their preferences with numerical values, called ratings. For example, ratings could be the number of stars assigned to a product or the indication if they liked or not a movie. Therefore, the utility function R is used by the system to calculate the ratings that users would probably assign to unknown items. Such a value measures the appropriateness of recommending an item to a certain user.

The ratings already available can be represented as a matrix, similar to the one illustrated in Table 2.2. In real systems, this matrix should be sparse, because the number of unknown cells is usually extremely elevated [112]. In fact, users can provide ratings on a limited set of items, especially if the catalog is very large.

	Item_1	Item_2	Item_3	Item_4	Item_5
User_1	5	3	4	4	?
User_2	?	1	?	3	3
User_3	4	3	?	?	5
User_4	3	?	1	?	?
User_5	?	5	5	2	1

Table 2.2: An example of rating matrix. Unknown ratings are denoted with an interrogative mark.

Different approaches can be exploited to compute unknown ratings. For example, in a collaborative filtering setting, popular techniques include user-based or item-based k-NN and matrix factorization methods [69].

Even if the recommendation problem is traditionally formalized as the task of predicting unknown ratings, many commercial systems provide to each user a ranked list of suggestions [81]. Therefore, a recommender system capable of selecting k items per user can be defined as a top-k recommender. It is straightforward to obtain a top-k list starting from the predicted ratings, as it is sufficient to sort them and select only the items with the highest computed values [2].

2.1.2 Sequential Recommender Systems

A recommender system designed to consider the timestamp associated to user ratings can be defined as a sequential recommender [108]. In the following, we review some examples of sequential recommenders available in literature.

Zhou et al. [140] proposed a web recommender system based on a sequential pattern mining algorithm. The recommender is trained with the access logs of a website and its goal is to predict the pages that are likely to be visited by a certain user, given her previously visited pages. The authors proposed to store the model in a tree-like structure, relying on a technique originally designed for matching substrings over a finite alphabet of characters. A recommendation is then created by matching the sequence of pages already visited by the target user with the sequences previously analyzed by the algorithm.

In the context of market basket analysis, it is also possible to exploit the sequence of previous transactions to predict what a customer is going to buy next [5]. Rendle et al. [110] proposed a method based on personalized transition graphs over Markov chains, while Wang et al. [134] designed a recommender capable of modeling both the sequential information from previous purchases and the overall preferences by a hybrid representation.

Bellogín and Sánchez [15] proposed a similarity metric designed to compare users in the context of CF recommender systems. This metric takes into account the temporal sequence of users' ratings to identify common behaviors. The authors argue that it is possible to consider a sequence of items as a string, where each character represents an item, and compare them using the longest common subsequence algorithm [63].

More recently, He et al. [59] introduced the concept of *translation-based* recommendation. While a traditional recommender only considers the pairwise interactions between items and users, their idea is to model a third-order relationship among a user, the items she interacted with in the past, and the item she is going to visit next. Each user can be represented as a vector in a transition space: given the current item, it is possible to compute where the next one will be located. At recommendation time, it is possible to generate a list of suggested items by relying on a nearest-neighbor search.

On the other hand, the task of generating recommendations of sequences was already discussed and presented in a seminal work by Herlocker et al. [62]. The authors suggested that it would be intriguing to be able to suggest, in the music domain, not only the songs that will be probably liked by a certain user, but also a playlist of songs that is globally pleasing. Moreover, they also proposed to apply this recommendation methodology in the context of scientific literature, where it is necessary to read a sequence of articles to become familiar with a certain topic.

The problem of recommending music songs was later addressed by Chen et al. [28], who designed and implemented a recommender system capable of generating personalized playlists by modeling them as Markov chains. Their algorithm is capable of learning, from a set of training playlists, how to represent each song as a point in a latent space. Then, starting from a seed song, it is possible to create a playlist of an arbitrary length by repeatedly sampling the transition probabilities

between adjacent songs. The resulting playlist is personalized because of the chosen seed. Furthermore, each user can influence the generation process by specifying some parameters: for example, a user might be more interested in popular songs, while another one in songs that are strictly related to the given seed.

Another typical application for a sequence-based recommender is the *next* point-of-interest (POI) prediction problem [108]. Given some training sequences of previously liked geographical locations, this task consists of predicting a sequence of venues that is pleasing for a given user. Feng et al. [43] proposed an algorithm capable of creating sequences of POIs that have not been already visited. The authors developed a Metric Embedding algorithm that captures both the sequential information and individual preference. Such metric is then exploited to create a Markov chain model capable of representing the transition probabilities between a given POI and the next one. The key features that are implicitly considered in the embedding creation phase are the conceptual similarity and the geographical distance of the analyzed venues.

A different line of research is represented by recommenders capable of analyzing sequences of multimedia objects. For example, Albanese et al. [7] proposed a hybrid recommender system for retrieving multimedia content based on the theory of social choice and capable of exploiting, among other signals, the implicit browsing preferences of its users. Later, the authors of [8] introduced a multimedia recommendation algorithm capable of combining semantic descriptors and usage patterns. The proposed approach can manage different media types and it enables users to explore several multimedia channels at the same time. Possible applications of such technologies are represented by browsing tools for virtual museums [10] and recommenders of cultural heritage sites [123].

2.1.3 Review-based Recommender Systems

The exploitation of user reviews in recommender systems is a well-known research topic, as reported by Cheng et al. [27]. Some techniques try to tackle the problem of building the profile of users by analyzing their reviews, while others focus on the identification of the main features of the items to recommend.

Different strategies have been proposed in the literature to address the latter problem. Some researchers have suggested methods able to identify the sentiment associated with the features of an item exploiting a domain-specific ontology [1] or its technical description [136]. A common aspect of these techniques is that the possible features are already available before performing the analysis. However, in literature there are also approaches for unsupervised extraction of product features and sentiment from reviews [107, 119].

Another possibility is to identify the main characteristics of an item with the help of natural language processing methods, without any previous knowledge of the context. For example, a popular technique considers bigrams that frequently

occur in reviews and that are associated with a word expressing an emotion [41]. In this case, the goal of the recommender system is suggesting items with the same features of the ones liked by the target user, but with a better global sentiment. In the best of our knowledge, there is only one attempt to exploit user reviews for recommendation tasks using semantic annotation. Dzikowski et al. [42] applied semantic annotation to reviews while users are editing them. Their goal was to produce annotated reviews of restaurants through Linked Data in order to generate tags to be associated with the reviewed items.

2.1.4 Linked Data-based Recommender Systems

In the past, some studies reviewed different Linked Data-based recommender systems that were proposed in literature [37, 44]. Typically, these recommender systems consider the relationships among resources by taking into account the existing links in the Web of Data and use them to measure a semantic similarity. Such relationships can be direct links or paths between the items to recommend. In the following, we summarize the main works in this field.

Damljanovic et al. [36] suggested domain experts in an open innovation scenario. Their approach generates recommendations by discovering related resources through hierarchical or transversal relationships in DBpedia. Passant [104] presented *dbrec*, a music recommender system, which mainly relies on a measure named Linked Data Semantic Distance (LDSD). This measure is based on the number of direct and indirect links between two resources. Heitmann and Hayes [60] also proposed a recommender system which exploited Linked Data to mitigate the new-user, new-item and sparsity problems of collaborative recommender systems.

More recently, Musto et al. [95] studied the impact of the knowledge available in the Web of Data on the overall performance of a graph-based recommendation algorithm. Vagliano et al. [131] presented a recommendation algorithm based on Linked Data which exploits existing relationships between resources by dynamically analyzing both their categories and their explicit references to other resources. Di Noia et al. [38] described a model-based approach to provide content-based recommendations with Linked Data. Ostuni et al. [98] defined a neighborhood-based graph kernel for matching graph-based item representations. Di Noia et al. [39] introduced SPrank, a hybrid algorithm which extracts semantic path-based features from DBpedia and computes recommendations using Learning to Rank.

2.2 Offline Evaluation

To the best of our knowledge, the first survey that deals with the problem of evaluating a recommender system was conducted by Herlocker et al. [62]. In their work, the authors discuss when it is appropriate to perform an offline evaluation

and when it is necessary to carry on an online, or in vivo, experiment. The former is particularly useful to select a small set of potentially good candidates that will be further compared in a real scenario. However, to be complete and trustworthy, such an evaluation needs to rely on a set of well-defined metrics, that should be able to capture all characteristics of the recommended items.

An offline analysis is an experimental approach based on user preferences collected before the introduction of the system under evaluation. Therefore, it is a simple and effective method to conduct large scale evaluations, usually considering different algorithms and datasets. Its major requirement is the availability of a collection of user preferences in the domain of interest. If the dataset at disposal also includes timestamps, it is possible to exploit it considering ratings and suggestions according to their temporal order. However, offline evaluations are based on datasets that are usually sparse: for this reason, it is not possible to reliably evaluate recommendations that involve items with no ratings available from the target user. Furthermore, offline experiments based on existing rating datasets cannot consider other important factors such as the usability of the user interface. An alternative approach is represented by live experiments. They can be classified as user studies when they are conducted in a controlled environment with a limited set of subjects or as online analyses when a recommender system is made available to a potentially large community of users.

In the following, we introduce the problem of experimental reproducibility, then we explain why it is necessary to consider a multicriteria set of metrics for conducting a reliable evaluation. Finally, we review existing visualization and generative approaches in the context of rating datasets.

2.2.1 Experimental Reproducibility

Different authors analyzed the experimental reproducibility of offline evaluations in the context of recommender systems. For example, Jannach et al. [70] compared several recommendation algorithms in an offline experiment, analyzing their performance by relying on a comprehensive evaluation framework. The authors considered different splitting protocols and metrics, designed to characterize both the accuracy, in terms of rating and ranking, and the coverage of the suggested items. The results of the experimental trails suggest that some common algorithms, despite their high accuracy, tend to only recommend popular items that are probably not very interesting for the users of a real system. This problem is related to the popularity biases introduced by the offline evaluation protocol: for this reason, it is not advisable to compare different algorithms by relying only on measures related to their accuracy. In addition, different splitting protocols produce significantly different and non-comparable outcomes.

Gunawardana and Shani [55] proposed a set of general guidelines for designing

experiments with the purpose of evaluating recommender systems. Such experiments can be classified as offline trails, user studies, or online analysis that involve a live system. Several properties of a recommender system can be evaluated: for example, the most common ones are user preference, prediction accuracy, coverage, and utility. The authors argue that the possibility of measuring these properties is strongly influenced by the kind of study and, in the most extreme scenario, some of them cannot be obtained. For example, it is very difficult to measure users' preference in an offline setting.

For each property, different commonly exploited metrics are presented and discussed. Even if the main metrics proposed for evaluating the most popular properties are widely known and understood, usually there is little agreement about the most appropriate metrics for characterizing the least common properties. For example, several definitions, and several metrics, related to the property of utility are available in the literature. The authors also point out that a key decision of offline experiment design is the splitting protocol because this choice will greatly influence the final outcome of the measures.

Bellogín et al. [14] proposed an evaluation framework designed following the methodologies of the information retrieval field. They suggest that the evaluation procedures available in information retrieval are widespread: for this reason, they could be successfully exploited by the recommender systems community to create a shared evaluation protocol based on ranking, as this setting is more similar to the one of a live system. Unfortunately, three different design decisions need to be taken to achieve this goal.

The items considered for the evaluation could be all items available in the dataset, or only the items available in the test set. The non-relevant items for a certain user could be represented by all items in the test set not rated by that user, or by a subset of it with a fixed size. Finally, the global metric could be computed by averaging its value on all users, or on all ratings available in the test set. Different design choices will result in different evaluation protocols and results. The authors also identify two sources of biases in offline evaluations protocols: the sparsity bias and the popularity bias.

Several software tools are available with the purpose of simplifying the process of comparing the performance of recommendation algorithms. They typically include some evaluation protocols and a reference implementation of well-known techniques. Said and Bellogín [116] compared several of these tools to check if their results are consistent. They discovered that the values obtained with the same dataset and algorithm may vary significantly among different frameworks. For this reason, it is not feasible to directly compare the scores reported by these tools, because they are obtained relying on several protocols. The discrepancies reported by the authors are mainly caused by the data splitting protocol, the strategy used to generate the candidate items, and the implementation choices related to the evaluation metrics.

2.2.2 Beyond Accuracy

In their survey dealing with the problem of evaluating a recommender system, Herlocker et al. [62] review several accuracy metrics usually exploited by different authors and they classify them into three categories: predictive accuracy metrics, classification accuracy metrics, and rank accuracy metrics. These groups are strictly related to the purpose of the recommender system: predicting a rating for each user-item pair, identifying an item as appropriate or not for a user, and creating an ordered list of recommended items for a user. After discussing the accuracy-based metrics, they argue that, in order to draw a reliable conclusion, it is necessary to also consider other properties of the recommended items. In their opinion, a recommender system should be capable of providing suggestions that are not only accurate but also useful. For example, an extremely popular item may be an accurate but not an interesting suggestion. For this reason, they also discuss other metrics that could be considered beyond the traditional concept of accuracy, such as coverage, learning rate, novelty, serendipity, and confidence.

The idea of relying not only on accuracy-based metrics is also supported by Ge et al. [47]. In their work, the authors state that the purpose of an evaluation protocol is to assess the quality of the recommended items and not their accuracy. However, metrics like precision and recall alone are not capable of verifying that the recommendations are actually useful. In fact, only the users of the system can judge their quality in the context of an online experiment. Therefore, their suggestion is to consider a multicriteria set of metrics and not only accuracy when it is necessary to perform an offline study.

2.2.3 Visualization Approaches

Different authors have proposed to create interactive visualizations for qualitative evaluating the goodness of the recommended items or helping the users to identify the most relevant suggestions. For example, Kunkel et al. [77] created a 3D map-based visualization that represents the preferences of a user on the entire space of items. The user can inspect the profile created by the recommender and also manually modify it, if necessary.

Çoba et al. [31] extended the *rrecsys* library by adding to it graphical capabilities for performing an offline visual evaluation of different recommendation approaches with respect to the popularity of the suggested items. Gil et al. [50] introduced VisualRS, a tool capable of creating tree graph structures for exploring the most important relationships between items or users. The graph-based visualization is useful for comparing the results of different recommendation approaches and selecting the most appropriate one for a given task. In contrast, Cardoso et al. [23] proposed to combine the output of different recommender systems with human-generated data to allow users to explore the suggested items in an effective way.

This method could also be exploited to compare the results of different recommender systems in a qualitative way.

2.2.4 Synthetic Datasets

Synthetic datasets are commonly used in literature to assess the performance of database systems or to study the behavior of data mining algorithms. For example, Agrawal et al. [6] created a generator of retail transactions intended for the evaluation of association rule algorithms, while Houkjær et al. [67] introduced a software capable of creating relational data for benchmarking purposes. Such tools can generate realistic data in terms of their statistical distributions, which can be empirically learned for existing datasets or provided by a researcher using domain-specific languages.

Similar approaches have been also explored in the field of recommender systems, usually because of the lack of public datasets with the required characteristics. Tso et al. [127] created a synthetic data generator for evaluating context-aware recommenders based on Dirichlet and Chi-square distributions. The metric of information entropy is then exploited to control the randomness of the synthetic data. A similar method has been discussed by Pasinato et al. [103]: their intuition is to represent the heterogeneous rating behaviors of the users with different statistical distributions.

Manouselis et al. [84] presented a tool, named CollaFis, capable of creating synthetic ratings for the evaluation of either single-criteria or multi-criteria recommender systems. The users of CollaFis need to specify the characteristics of the generated data, like the number of users, items, and criteria. A common aspect of all the previously mentioned methods is that researchers are required to choose and configure the statistical distributions that are exploited to generate the artificial datasets. However, the main problem of such an approach is that it is impossible to predict the real behavior of many different users with a few statistical distributions [88].

Another possible line of research is related to the imitation of a real collection of preferences. For example, Rodríguez-Hernández et al. [25] developed a software, DataGenCARS, for creating artificial ratings using a set of parameters provided by the user or inferred from a reference dataset. However, in Chapter 7, we argue that statistics computed at a global level are not informative enough to create a synthetic dataset, as they are not able to capture the different behaviors of the various groups of users.

Chapter 3

Multicriteria Recommender Systems

The traditional approach to the recommendation problem discussed in Section 2.1 could be considered somewhat limited, as users typically tends to judge items according to different criteria [2]. For example, we can easily imagine to assign different ratings to a movie, expressing how much we liked the story, the acting, the direction, and the visual effects. Such multiple ratings could be exploited by a recommender system in order to identify more effectively which items should be suggested. For this reason, different authors started to propose multicriteria recommender systems, namely methods capable of suggesting items by relying on ratings provided over different criteria instead of a single one [4, 83].

In this chapter, we investigate the state of the art in the field of multicriteria recommender systems. We follow the systematic literature review protocol proposed by Kitchenham & Charters [76], in order to enable other researchers to easily verify and reproduce our work. We consider nine different research questions that encompass various aspects of the reviewed studies.

In particular, we analyze the most important problems that multicriteria recommenders aim to address, as well as the exploited recommendation approaches, according to the taxonomy created by Burke [19]. We also describe the different machine learning and data mining techniques typically included in a multicriteria recommender and we identify which methods are frequently utilized in each recommendation phase, thus we try to describe the structure of an ideal multicriteria RS. We quantitatively measure the domains that are the most appropriate ones for such systems and we review how the proposed algorithms have been evaluated with respect to the experimental settings, the metrics, and the exploited datasets. Finally, we describe the most promising directions for future works that are mentioned in the reviewed studies.

We considered a total number of 93 studies, published from 2003 to 2018, to perform this systematic literature review. To the best of our knowledge, this is

the first review conducted in the field of multicriteria recommender systems that follows a standardized and repeatable protocol. We aim that our study could be useful to other researchers working in this area, especially for better identifying possible approaches and future trends.

The remainder of this chapter is structured as follows. In Section 3.1, we detail the protocol that we followed for conducting the review. Then, we present the quantitative results in Section 3.2 and we provide a possible interpretation of the outcomes of the review in Section 3.3. Finally, we conclude this chapter with Section 3.4, while, in Appendix A, we report the list of selected studies.

3.1 Methodology

We decided to perform this review according to the guidelines designed by Kitchenham & Charters for Systematic Literature Reviews (SLR) in the field of Software Engineering [76]. This method guarantees that the outcome of the review is verifiable and repeatable by other researches. The protocol, which is graphically illustrated in Figure 3.1, was developed by the author of this dissertation.

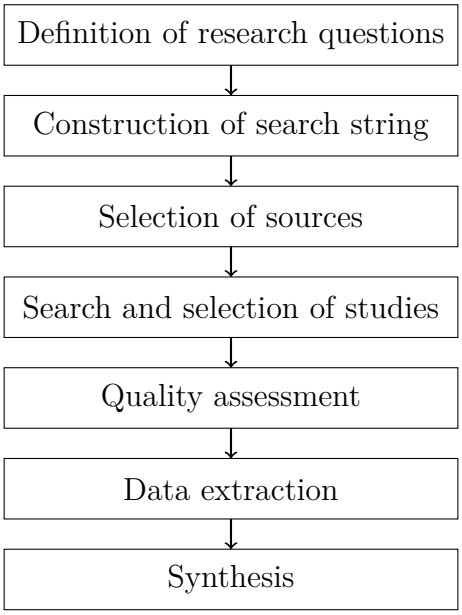


Figure 3.1: The systematic literature review protocol designed following the guidelines by Kitchenham & Charters [76].

3.1.1 Research Questions and Search String

The purpose of this systematic literature review is to identify the studies describing multicriteria recommender systems and to understand the motivations behind

their usage, the techniques employed, the experimental protocols used to validate them, and the related research challenges. For these reasons, we defined the following research questions.

RQ1.1 What are the most relevant studies addressing multicriteria RSs?

RQ1.2 What are the most challenging problems faced by researchers?

RQ1.3 What are the approaches used by multicriteria recommenders?

RQ1.4 Which techniques and methods have been proposed?

RQ1.5 In which domains multicriteria recommender systems are applied?

RQ1.6 Which protocols and frameworks are used for their evaluation?

RQ1.7 Which metrics are considered during their evaluation?

RQ1.8 Which datasets are used for testing the algorithms?

RQ1.9 What are the most promising directions for future works?

In order to retrieve the studies related to multicriteria recommender systems, we defined the following preliminary set of keywords: {Multicriteria, Recommender System}. This initial set was expanded to include alternative spellings and we defined the search string used to query the digital sources as follows.

```
(multicriteria OR "multi criteria" OR "multi-criteria")
AND ("recommender system" OR "recommendation system")
```

We selected six scientific digital libraries that contain primary studies related to the field of computer science, as detailed in Table 3.1. Other more general sources, like Google Scholar, were not included because they usually index studies already available in the primary sources.

Source	URL
ACM Digital Library	https://dl.acm.org
IEEE Xplore	http://ieeexplore.ieee.org
ISI Web of Knowledge	http://www.webofknowledge.com
ScienceDirect	https://www.sciencedirect.com
Scopus	https://www.scopus.com
Springer Link	https://link.springer.com

Table 3.1: The digital libraries considered during the search process.

3.1.2 Selection Process

The selection process was performed during January 2019. We inserted the search query in the search field of the digital libraries selected as sources for the review and we retrieved all the studies identified by the respective search engines. Because of the high number of false positive results, we decided to limit our search to the title, abstract, and keywords with IEEE Xplore, ISI Web of Knowledge, and Scopus. The preliminary set initially contained 1256 studies. We checked their titles and authors in order to discover possible duplicates: after having removed duplicated results, the preliminary set was reduced to 950 studies.

Furthermore, for objectively identifying the studies to include in the review, we defined a set of inclusion and exclusion criteria, which are summarized in Table 3.2. We first applied the criteria in a coarse selection phase by only considering their abstracts and we obtained a list of 301 papers. Then, we analyzed again the available studies in a detailed selection phase by reading relevant portions of their content. We finally selected 93 studies as part of this literature review, as summarized in Table 3.3. The full list, sorted by source, year of publication and author, is available in Appendix A.

Code	Inclusion criteria
IC1	Papers describing multicriteria recommender systems
IC2	Papers published in conferences and journals
IC3	Papers written in English language
Code	**Exclusion criteria**
EC1	Papers not addressing recommender systems
EC2	Papers addressing RSs without multicriteria ratings
EC3	Papers that report only abstracts or posters
EC4	Papers that describe a planned research[1]
EC5	Grey literature and book chapters

Table 3.2: The inclusion and exclusion criteria.

3.1.3 Quality Assessment

For objectively assessing the quality of the studies selected as part of this review, we defined eight quality questions, as listed in Table 3.4. It is possible to assign to each question the scores of 0, 0.5, and 1, that correspond, respectively, to the

[1]We defined a planned research as a study that only contains a high level description of the proposed methodology, without the details necessary to implement it.

Source	Search	Coarse	Detailed
ACM Digital Library	27	24	11
IEEE Xplore	38	36	15
ISI Web of Knowledge	31	25	6
ScienceDirect	344	81	19
Scopus	118	62	24
Springer Link	392	73	18
Total	950	301	93

Table 3.3: The number of studies after each selection step.

answers *Yes*, *Partly*, and *No*. During the quality assessment phase, we provided an answer to each question for all studies included in the review.

Code	Quality question
QQ1	Did the study clearly describe the problems that it is addressing?
QQ2	Did the study review the related work for the problem?
QQ3	Did the study compare its approach with possible alternatives?
QQ4	Did the study describe the components of the proposed RS?
QQ5	Did the study provide an empirical evaluation of the solution?
QQ6	Did the study present a clear statement of the findings?
QQ7	Did the study analyze the application scenarios of the RS?
QQ8	Did the study recommend any further research activity?

Table 3.4: The quality questions.

3.1.4 Data Extraction

We carefully read multiple times the primary studies that are selected as part of this review. During this phase, we identified the data available in the works useful for providing an answer to the research questions introduced in Section 3.1.1. More in details, we looked for the information listed in Table 3.5. This process was supported by the data analysis software tool NVivo.[2] We relied on this tool to minimize the manual effort required for applying the methodology described in Section 3.1.5.

[2] https://www.qsrinternational.com/nvivo

Field	Description	RQ
Code	An internal identifier of the study	-
Title	-	RQ1.1
Authors	-	-
Publication year	-	RQ1.1
Publication name	-	-
Source	The digital library that contains the study	-
Type	Conference or journal	-
DOI	-	-
Research problem	The problem that the study tries to address	RQ1.2
Contribution	The description of the proposed method	RQ1.3
Implementation	How the method was implemented	RQ1.4
Domain	The domain of the recommended items	RQ1.5
Evaluation protocol	The protocol used to evaluate the method	RQ1.6
Evaluation metric	The metric used to compare the RS	RQ1.7
Dataset	The dataset used to execute the evaluation	RQ1.8
Limitation	The limitations of the proposed method	RQ1.9
Future work	The suggestions for future works	RQ1.9
Quality score	-	-

Table 3.5: The data extraction form.

3.1.5 Synthesis

We synthesized the results of our review following the Cruzes & Dyba methodology [34] for combining and comparing the results of the primary studies that we considered. While reading the selected studies, we associated relevant portions of their text with codes. A code is a label applied to text segments that discuss the same theoretical or descriptive idea and that is used to aggregate in an organic way the data that we are analyzing. We initially defined some general codes associated with the research questions. Then, we created more specialized sub-codes related to the content of the studies, thus following an integrated approach that combines both inductive and deductive methods and that is considered the most appropriate one for a systematic review [34]. We subsequently aggregated the codes in themes, and we mapped these themes back to the original research questions. The outcomes of this last phase are reported in Section 3.2, grouped by research question.

3.2 Results

In this section, we highlight the findings of our systematic literature review regarding multicriteria recommender systems, according to the research questions

introduced in Section 3.1.1. These results will be further discussed in Section 3.3.

3.2.1 Included Studies

The main purpose of RQ1.1 is to identify the studies related to the topic of multicriteria recommender systems to be included in this review. Following the protocol detailed in Section 3.1, we identified a total number of 93 studies. These works have been presented during conferences or they have been published in scientific journals in a period of time from 2003 to 2018. In Figure 3.2, we detail the number of studies per year and per venue. It is possible to observe an increasing amount of studies published in the last years.[3]

We exploited the questions listed in Table 3.4 to assess the quality of the works included in this review. In Figure 3.3, we report the quality scores according to the publication venue. As expected, journal papers obtained, in general, higher scores with respect to conference papers. Furthermore, Figure 3.4 contains the average quality scores for each quality question. It is possible to observe that the highest scores are associated with QQ1 (*Did the study clearly describe the problems that it is addressing*?), while the lowest ones with QQ8 (*Did the study recommend any further research activity?*).

3.2.2 Research Problems

In this section, we describe the main problems and challenges that multicriteria recommender systems aim to address and, thus, we provide an answer to RQ1.2. In total, we identified 10 different categories of problems that are mentioned in the reviewed studies. The number of studies for each category is summarized in Figure 3.5. It is important to observe that a single study may analyze different problems at the same time.

Data Sparsity

Data sparsity is the most frequent problem in this field and it is caused by the fact that users provide ratings for a limited number of items or criteria. While this is a well documented common issue of recommender systems, multicriteria user-item matrices may be even sparser, as they require more effort and time from the users of the system. In order to address this problem, several solutions are proposed in the reviewed studies. For example, the authors of [P3] suggest to combine the multicriteria ratings using two different regression functions, one for the items and

[3]Please note that the results for the year 2019 are not available, as the selection was performed in January 2019.

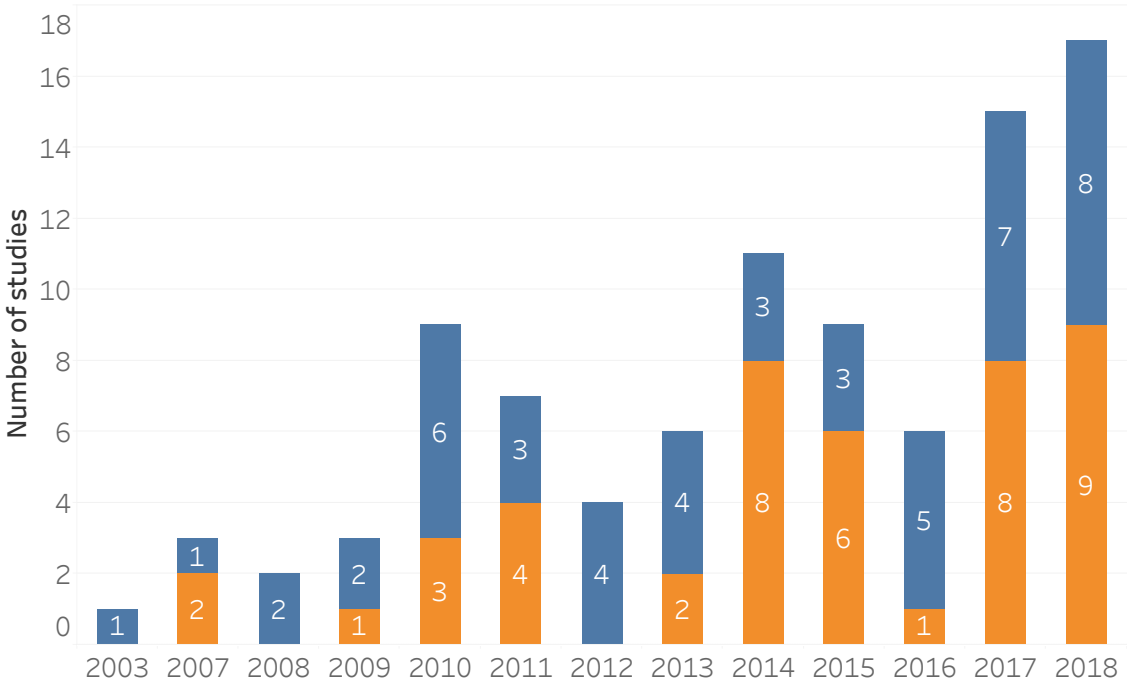


Figure 3.2: This stacked barplot represents the number of included studies per year of publication. Blue studies have been presented in a conference, while orange studies have been published in a journal.

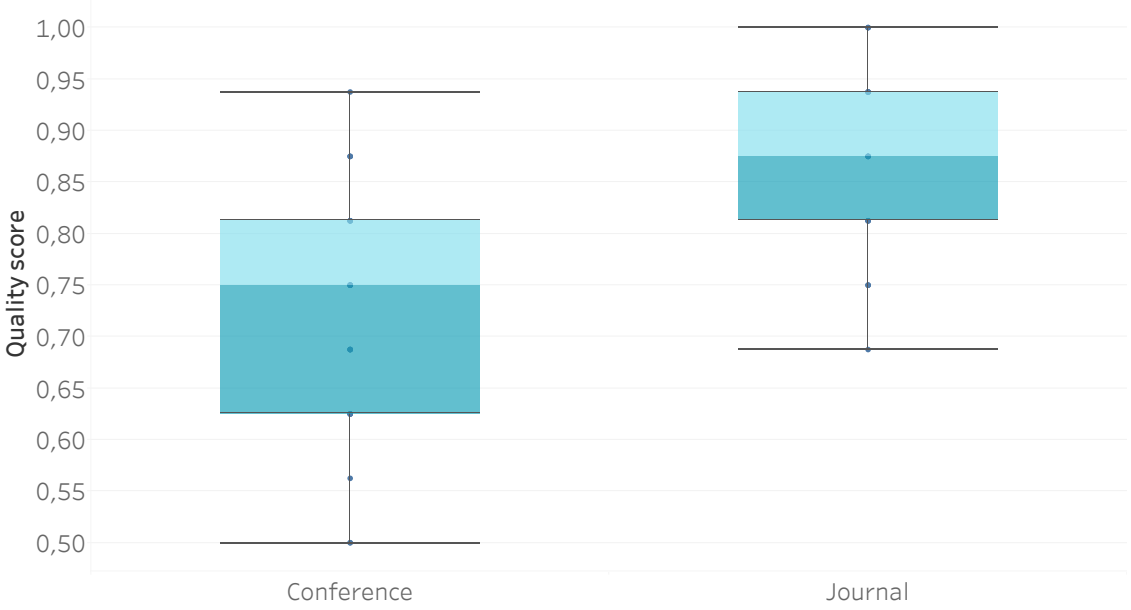


Figure 3.3: This boxplot represents the distribution of quality scores per publication type. Studies published in journals have higher quality scores.

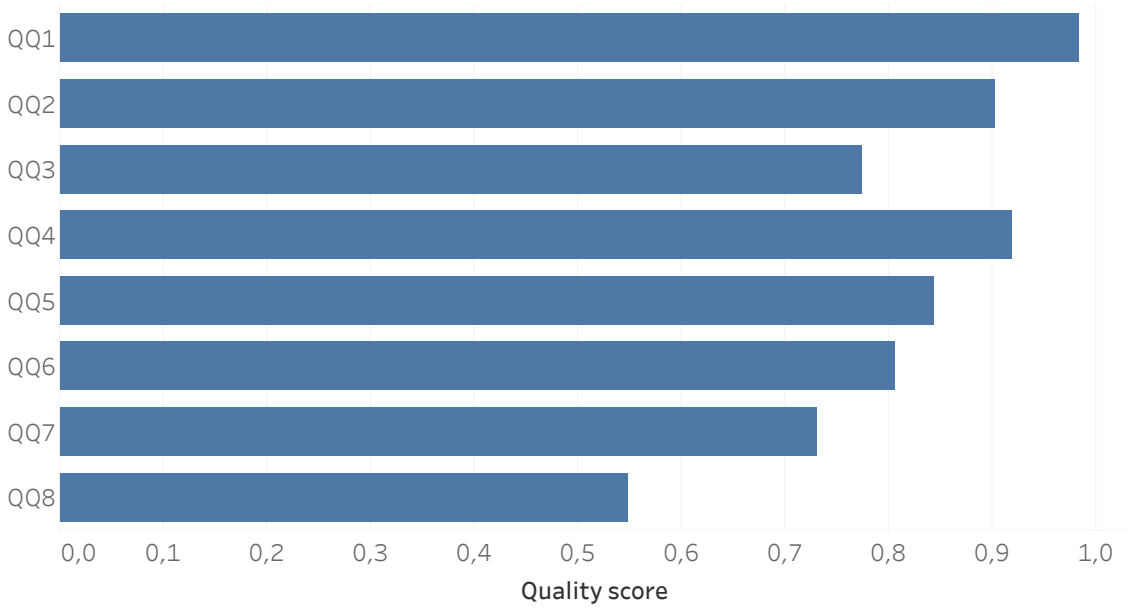


Figure 3.4: This barplot represents the average quality score per quality question. The lowest quality score is associated with the description of future works.

one for the users. The ratings estimated by the regression functions are then combined in order to minimize the prediction error. In [P18], a Bayesian latent model for multicriteria recommenders is exploited along with a support vector regression learner in order to mitigate the data sparsity problem. Another possible solution, as suggested in [P63], is represented by dimensionality reduction techniques, which can be used to obtain a more compact representation of each user. Finally, it is possible to integrate the available ratings with an external ontology [P14] or with a trust-based model [P72]. The data sparsity problem was identified in a total number of 22 studies.

Criteria Weights

In order to provide accurate suggestions, it is of paramount importance being able to discover the relationships among the different criteria and to identify the most relevant ones for the target user. A wide range of possible solutions is available in the analyzed studies. For instance, the authors of [P8] identify the most important criteria for a user exploiting a statistical technique based on the average ratings of each item. Other studies analyze several machine learning methods. In [P9], the author proposes to consider *chains* of criteria instead of exploiting all criteria together: the rating on each criterion is estimated considering the previous predictions as context information. In [P5], the optimal weights are learned using particle swarm optimization, while in [P93] an artificial neural network is exploited

for this purpose. Another popular solution is represented by decision making methods. As an example, in [P51] users are asked to perform pair-wise comparisons of the available criteria. The problem of selecting proper criteria weights was explicitly mentioned in 17 studies.

Personalization

Any recommender system should be capable of suggesting items that match the preferences of the target user. The possibility of exploiting multiple ratings for each item is often considered an effective way of increasing the accuracy of the recommendations, for example this fact is mentioned in [P15] and [P32]. It is also important being able to understand what are the most relevant criteria for each user, as suggested by the authors of [P8]. On the other hand, it is essential to avoid including criteria that are redundant, as this may negatively affect the performance of the system [P29]. In [P48], a multicriteria recommender system is combined with a content-based approach in order to generate better suggestions, while the authors of [P10] propose to increase the accuracy of the recommendations using a deep learning technique called Stacked Autoencoders. In general, this problem was explicitly considered by 15 studies.

Data Noise

The presence of noise in data is typically related to the fact that users may provide ratings that are biased or even dishonest. For example, users may not understand the meaning of each criterion and may find difficult to express their preferences on a numerical scale, or may be bored by the request of assigning many ratings to a single item. In [P89], fuzzy logic techniques are exploited in order to address the uncertainty of user preferences, while in [P72] such techniques are combined with a trust-based model. The authors of [P3] propose to mitigate this problem by performing feature selection in a pre-processing step, where the most relevant criteria of a certain dataset are identified. Another possible solution to this problem is represented by the idea of considering the numerical differences between ratings instead of their absolute values in the recommendation process [P32]. Data noise was addressed by 14 studies.

Cold-start

The cold-start problem is a well-known issue in the field of recommender systems based on collaborative filtering approaches. It can be defined as the impossibility of creating reliable suggestions due to the lack of data regarding a new user or a new item. The authors of [P60] aim to solve it by providing non-personalized recommendations to new users and exploiting content-based features when a new item is added to the system. A different approach is represented by the elicitation

of user preferences using decision making techniques and multicriteria ratings [P66]. In [P47], a multicriteria implicit feedback method based on user behavior analysis is discussed, while the authors of [P15] propose to tackle this issue with a trust-based model. As a last example, a knowledge-based method that is immune to the cold-start problem is illustrated in [P12]. Cold-start was considered a research issue in 14 studies.

Scalability

Scalability is a general problem of collaborative filtering recommender systems, especially for the ones developed in an academic context as proof-of-concept. Because many multicriteria recommenders require multiple runs of such algorithms, it is reasonable to suppose that scalability issues are even more widespread. For example, this issue is discussed by the authors of [P24], who describe a multicriteria recommender that exploits a distributed architecture based on Apache Spark. In [P32], a clustering algorithm is applied for creating groups of similar users that can be used to compute predictions in a scalable way. A popular dimensionality reduction technique discussed by [P19] and [P84] is the higher order single value decomposition. In total, 12 studies considered the scalability problem.

User's Effort

As multicriteria recommenders usually require many ratings for each user and item pair, explicit elicitation methods are intrusive and may waste the user's effort and time. For this reason, the authors of [P30] and of [P50] propose a multicriteria recommender based only on implicit feedback. Another possible solution is described in [P63], where their authors develop a hybrid profiling framework for reusing traditional ratings with multicriteria recommenders. A similar approach to this problem is presented in [P48], where multicriteria ratings are computed starting from single ratings and content-based information. This problem was discussed by 11 studies.

Other Research Problems

Other research problems are mentioned in a more limited number of studies. In particular, 4 studies analyzed the issues related to the selection of a proper similarity metric in the context of multidimensional neighborhood-based collaborative filtering, while 3 studies reported the challenges related to the execution of a reliable evaluation protocol. Finally, 3 more recent studies mentioned the problem of fairness in the selection of the recommended items, both with respect to the unique peculiarities of the users and to the characteristics of the catalog.

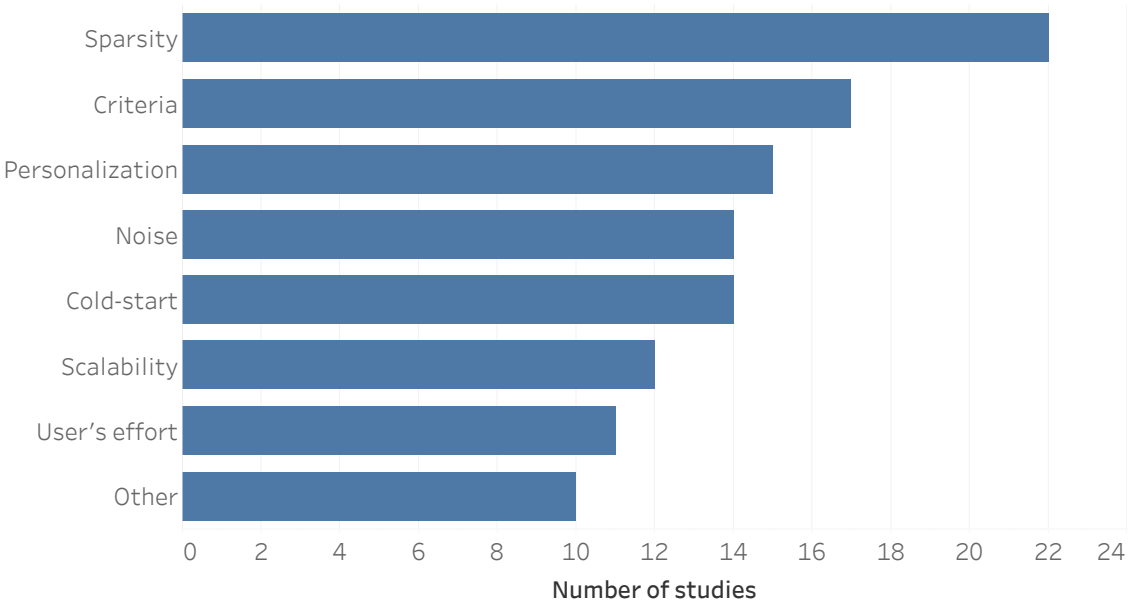


Figure 3.5: This barplot represents the number of studies per research problem.

3.2.3 Recommendation Approaches

In order to provide an answer to RQ1.3, we analyzed the studies included in this systematic literature review and we classified them according to the taxonomy provided by Burke [19]. This taxonomy has become a widespread way of characterizing different recommendation approaches. However, we decided not to consider hybrid recommender systems, as almost all multicriteria recommenders would fall in this category. For this reason, if a study combines multiple approaches, it will be included in all the categories of the different methods that are mentioned in the study. We summarize the number of available studies for each recommendation approach in Table 3.6.

Collaborative Filtering

Collaborative filtering is the most popular recommendation technique described in the reviewed literature. In a traditional recommender system, the users whose rating behaviour is similar to the one of the target user are exploited for selecting the items to be suggested. In a multicriteria recommender, a popular approach consists in applying collaborative filtering algorithms on the available criteria and then combining the results in a global estimated rating, like in [P13] and [P28]. An alternative to the user-based approach is represented by the item-based collaborative filtering, where the similarity is computed among the items, as discussed, for example, in [P65]. The authors of [P3] and [P45] describe how to combine user-based and item-based models in a comprehensive approach. Collaborative filtering

may also be implemented with a model-based approach, for instance matrix factorization (e.g., in [P21] and [P75]). In general, this technique may be combined with other methods, like content-based approaches [P16], clustering [P32], and fuzzy logic [P60]. In total, collaborative filtering was exploited by 82 studies.

Content-based

A content-based recommender system considers the previous preferences of a user in order to build a profile and to select items with similar characteristics. For example, the authors of [P79] describe a recommender system of research papers that relies on content-based and multicriteria collaborative filtering algorithms. In [P43], user profiles are created from multicriteria ratings and, then, they are exploited for building clusters of similar users. The authors of [P30] suggest to identify the user's category-wise preferences for each criterion with a content-based approach. A possible source of structured information regarding items are external ontologies, as they are discussed in [P16]. In [P6] and [P20], user's reviews are mined to identify the most important features of each item. Content-based approaches were mentioned by 16 studies.

Knowledge-based

A knowledge-based recommender system relies on an externally encoded domain knowledge in order to match the user profiles with certain item features. For example, in [P69] a mobile recommender suggests restaurants considering their geographical location and cuisine. The author of [P46] describes a method for asking users to express their preferences regarding the features of a smartphone. In a similar vein, the recommender system presented in [P76] exploits user profiles and fuzzy set theory in order to suggest cities to be visited. In [P47], the authors of the study propose a set of rules for building a personalized list of recommended movies. Knowledge-based techniques were identified in 12 studies.

Community-based

If a recommender system also considers the relations of friendship and trust among its users, it is defined as community-based. For example, the authors of [P72] propose a multicriteria collaborative filtering recommender that is enhanced by considering trust as an additional weight during the hybrid prediction phase. A similar approach is followed by [P61] and [P91], where the trust score for each user is computed only considering rating data and it is combined with the results of a collaborative filtering algorithm. Community-based recommenders were discussed by 5 studies.

Demographic

A demographic recommender considers the demographic profile of the user for suggesting items. For example, the authors of [P31] describe a multicriteria recommender for groups where users are clustered also according to their demographic profile. The other studies that mention demographic information are [P45], [P43], and [P11]. Demographic recommenders were considered by 4 studies.

Approach	Studies
Collaborative filtering	82
Content-based	16
Knowledge-based	12
Community-based	5
Demographic	4

Table 3.6: The number of studies per recommendation approach.

3.2.4 Multicriteria Techniques

In the following, we analyze the main techniques and methods related to multicriteria recommender systems that we identified in the reviewed studies and, therefore, we provide an answer to RQ1.4. In most studies, the recommenders combine different techniques, for example k-NN may be exploited to estimate unknown ratings, while decision analysis to merge the different ratings in a global prediction. We summarize the most frequent ones in Table 3.7.

k-NN

k-NN is a classification method used in data mining applications that relies on the similarity of the instances to be classified with the training examples. In the context of collaborative filtering, k-NN is exploited to find similar users or items considering their neighborhood. For example, in [P13], a k-NN collaborative filtering method is applied to each criterion separately, like in traditional recommenders. In contrast, a different approach to this problem is considering all the criteria together when applying the k-NN algorithm. To this end, it is necessary to rely on a multidimensional distance, like the Manhattan, Euclidean, or Chebyshev distance, as described in [P53]. k-NN is the most popular data mining technique applied to multicriteria recommender systems, as it was identified in 35 studies.

Decision Analysis

In order to rank items with contrasting criteria, it is possible to exploit the tools provided by multiple-criteria decision analysis, which is a sub-field of operations research. In general, different methods are available in order to support users in making complex decisions, and some of these methods have also been applied to multicriteria recommender systems. For example, in [P38], an analysis hierarchy process (AHP) is used in order to help users to evaluate the relative importance of each criterion. In [P59], the UTA* algorithm is exploited for constructing user profiles that are subsequently grouped according to their preferences. Other decision analysis methods mentioned in the reviewed studies are, for instance, ELECTRE [P37], SMART [P35], TOPSIS [P31], and UTADIS [P78]. In total, decision analysis techniques were identified in 26 studies.

Fuzzy Logic

Fuzzy logic is a mathematical model that can be used to represent the concept of partial truth. This model is exploited to formalize the vagueness and uncertainty that are usually associated with user ratings. For example, in [P17], [P60], and [P25], ratings are expressed using linguistic terms in a qualitative way, considering that each term may have a different meaning according to the user. Furthermore, the authors of [P69] use the AHP decision analysis method in the fuzzy domain using fuzzy numbers instead of real numbers. In [P84], fuzzy rules that express how to build global ratings are identified for each cluster of users. Fuzzy logic was exploited as a recommendation technique in 16 studies.

Regression Analysis

Regression analysis is a set of statistical techniques for predicting the value of a dependent variable given one or more independent variables. In the reviewed studies, such techniques are typically used to estimate the global rating of an item considering the predicted ratings for each criterion. For example, the authors of [P75] find the weights of the aggregation function with a linear regression model that is learned for each user. In [P88], a non-personalized linear regression model is first used to aggregate the similarities among users, and then to estimate the final ratings. A different approach is represented by Support Vector Regression (SVR): for instance, in [P82], a SVR model is trained for each user in order to synthesize the overall rating. We found regression analysis techniques in 15 studies.

Clustering

Clustering is an exploratory data mining approach that consists in grouping objects in cohesive sets. A typical application of such techniques to multicriteria recommender systems is represented by the identification of users with similar profiles. For instance, in [P32], [P59], and [P52], the global K-means clustering algorithm is exploited in order to create groups of users with similar preferences. In [P82], clusters of users are created according to the importance given to each criterion. On the other hand, the authors of [P56] propose to cluster the items and, then, to learn an aggregation function for each user and item cluster. A clustering technique is also exploited to identify malicious users in the context of robust recommenders [P27]. In total, clustering algorithms were mentioned in 15 studies.

Matrix Manipulation

In the reviewed studies, we identified different techniques used to compute predicted ratings with mathematical operations on matrices. For example, in [P90], the Singular Value Decomposition (SVD) method is exploited to compute unknown ratings for each criterion. In contrast, the authors of [P19] propose to reduce the dimensionality of the user, item, and criterion tensor with the Higher-Order Singular Value Decomposition (HOSVD) method and then to apply a collaborative filtering algorithm to the resulting matrix. In [P64], a matrix factorization technique is applied to a utility matrix estimated from the multicriteria ratings using a neural network model trained considering each user. A different approach is followed by the authors of [P92], which proposes a factorization machine model for representing all multicriteria ratings together. Matrix manipulation techniques were discussed in 12 studies.

Neural Networks

Neural networks are usually applied to multicriteria recommender systems in order to aggregate the predicted ratings for each criterion in a global score. For example, in [P24], a single layer PERCEPTRON algorithm is selected for this task, while the authors of [P73] propose a neural network trained with the simulated annealing algorithm. In [P84], an Adaptive Neuro-Fuzzy Inference System (ANFIS) is exploited for extracting fuzzy rules for each cluster of users; such rules are later applied to predict the overall rating. A different approach is described in [P93], where a neural factorization machine is used to model the interactions among users, items, and criteria, and in [P37], where a single layer PERCEPTRON is exploited to estimate the similarity among users. We identified neural network approaches in 12 studies.

Genetic Algorithms

Genetic algorithms can be considered a family of optimization techniques and, in the reviewed studies, they are typically used to determine the weights of each criterion. For example, in [P55] and [P87], a genetic algorithm is run for each user in order to construct a personalized aggregation function. In contrast, the authors of [P3] propose to use it for performing a feature selection of the available criteria in order to identify an optimal set of dimensions. In total, genetic algorithms were exploited by 7 studies.

Other Techniques

Other less frequent techniques described in the reviewed studies include statistical modeling [P18], particle swarm optimization [P5], and natural language processing [P36]. Such techniques were found in 7 studies.

Technique	Studies
k-NN	35
Decision analysis	26
Fuzzy logic	16
Regression analysis	15
Clustering	15
Matrix manipulation	12
Neural networks	12
Genetic algorithms	7
Other	7

Table 3.7: The number of studies per recommendation technique.

3.2.5 Application Domains

We analyzed the application domains of the multicriteria recommender systems described in the reviewed studies in order to provide an answer to RQ1.5. A graphical summary listing the categories of recommended items, considering possible examples and the experimental evaluation, is available in Figure 3.6. We observe that the majority of studies propose to apply multicriteria recommenders to domains related to tourism and travel. For example, 13 studies describe recommenders for hotels, 9 related to restaurants, and 4 dealing with tourist places. Another popular domain is related to movies, mentioned in 8 studies. Other domains include consumer electronics products and education, described in 7 and 5 studies respectively. Research papers were discussed in 3 studies, while medical

treatments and music in 2 studies each. Less popular domains, identified only in 1 study and grouped in a miscellaneous category, are business and romantic partners, investment solutions, electronic books, and job opportunities.

Different categories of criteria are selected by researchers according to the domain. For example, popular criteria for hotels are *rooms*, *location*, *cleanliness*, *service*; for restaurants *food quality*, *service*, *presentation*, *taste*; for tourist places *architectural style*, *ease of access* and *welcome quality*; for movies *story*, *acting*, *direction* and *visuals*; for consumer electronics products *type*, *brand*, *weight*, *size*; for learning resources *subject relevance* and *educational value*.

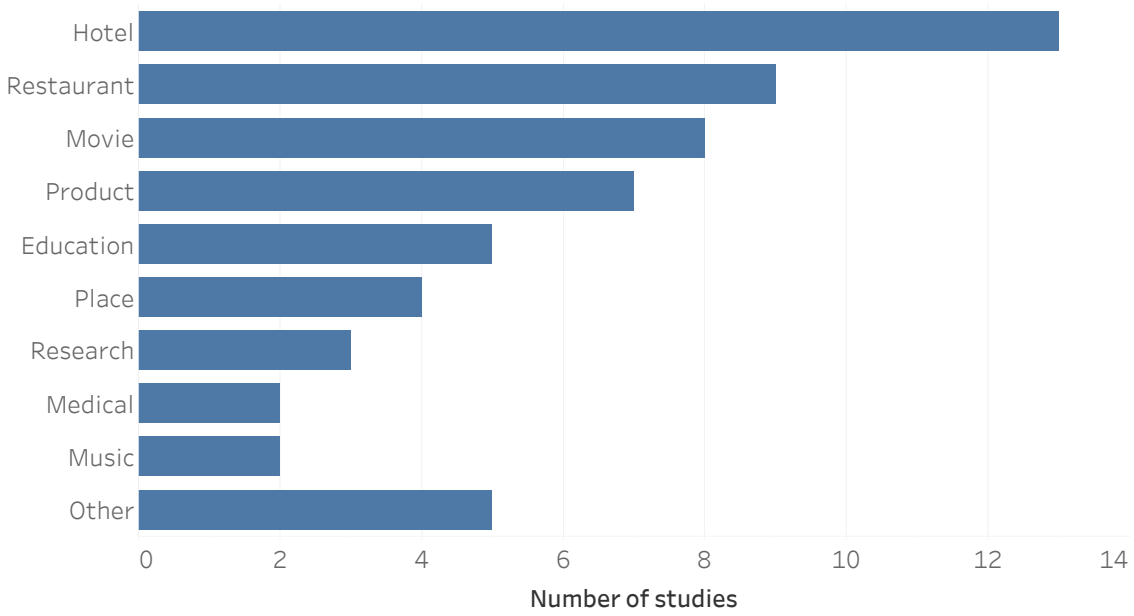


Figure 3.6: This barplot represents the number of studies per application domain.

3.2.6 Evaluation Protocols

In this section, we describe the evaluation protocols followed by the reviewed studies, in line with RQ1.6. We grouped the possible evaluation strategies in four main categories, which are summarized in Table 3.8.

Offline Comparison

We discovered that 74 studies compare the proposed solution with other approaches using an offline evaluation. Multicriteria recommender systems are usually compared against traditional baselines such as single-criteria recommender or weighted average multicriteria approaches. Different studies consider the most similar methods already available in literature, while few studies only compare the

described technique with itself, analyzing several configuration parameters.

User Study

A different approach is represented by the execution of user studies, which were carried out in 12 works. For example, the authors of [P47] created a movie recommender system that was tested by 567 users. The researchers computed different metrics considering their behaviour while utilizing the recommender. In contrast, in [P86] 158 users were asked to fill out a questionnaire in order to compare different recommendation models. User studies are also exploited to evaluate the usability of the system, as done, for instance, in [P58].

Case Study

We also identified 4 studies that evaluated the proposed approach by describing a case study. For example, in [P4], a possible application of a multicriteria recommender to the movie domain is discussed, while the authors of [P54] empirically compare the suggested restaurants considering different user profiles.

No Evaluation

Finally, 3 studies performed no evaluation of the multicriteria recommender presented in the paper. For example, in [P15], the evaluation of the proposed model is left as a future work.

Evaluation protocol	Studies
Offline comparison	74
User study	12
Case study	4
No evaluation	3

Table 3.8: The number of studies for each evaluation protocol.

3.2.7 Evaluation Metrics

In the following, we discuss the metrics exploited in the reviewed studies for conducting the experimental evaluation of the proposed solutions in order to answer to RQ1.7. We decided to classify them according to the dimensions related to the recommender system proprieties described by Gunawardana et al. [55]. The identified category for each evaluation metric and the associated number of studies are reported in Table 3.9.

Rating Accuracy

This category includes metrics designed to evaluate the capability of the system to correctly estimate user ratings. In particular, 51 studies report the Mean Absolute Error (MAE), 20 the Root Mean Squared Error (RMSE), and 3 the Mean Squared Error (MSE). Other metrics exploited by 1 study each are the coefficient of determination (R^2) and the Mean Absolute Percentage Error (MAPE). In total, rating accuracy metrics are mentioned in 59 studies.

Usage Accuracy

If the goal of a recommender is to predict a list of items, it is possible to evaluate the usage accuracy of the available suggestions. Precision is considered in 36 studies, recall in 28, F1 in 23 studies, and Area Under the Curve (AUC) in 4 studies. Usage accuracy is the second most popular category, as it was identified in 43 studies.

Coverage

The metric of coverage was computed in 11 studies. Even if this metric can be evaluated both at the level of users and at the level of items, all the studies included in this review considered the coverage of the item space, also known as catalog coverage [62].

Ranking Accuracy

The correctness of the ranking in the recommended lists of items was analyzed by 10 studies. In details, 8 studies exploit the Normalized Discounted Cumulative Gain (nDCG) metric, while 4 studies the Fraction of Concordant Pairs (FCP). A less popular metric, described by 1 study, is the Kendall's τ.

Scalability

The authors of 8 studies evaluated the scalability of the proposed approach. A typical metric used to this purpose is the time required to compute the predictions, which is reported by 6 studies. In contrast, 2 studies exploit the speed of the recommendations.

Other Metrics

Additional metrics identified in the reviewed studies include the utility of the suggested items and the system satisfaction, evaluated with a user study, and the robustness of the recommendations. Finally, the authors of [P68] defined a combined metric.

Evaluation metric	Studies
Rating accuracy	59
Usage accuracy	43
Coverage	11
Ranking accuracy	10
Scalability	8
Other	5

Table 3.9: The number of studies for each evaluation metric.

3.2.8 Evaluation Datasets

In line with RQ1.8, we analyzed the datasets exploited for conducting the experimental evaluation of the techniques described in the reviewed studies. In total, 74 studies mentioned at least one dataset: this result is consistent with the number of studies that performed an offline comparison, as reported in Section 3.2.6. We summarize the studies for each dataset in Figure 3.7.

Yahoo! Movies

Yahoo! Movies was a website, part of the Yahoo! network, that provided information and reviews about movies. Among other features, users were able to rate each movie considering five criteria: story, acting, direction, visuals, and overall. Yahoo! Research provides a public Yahoo! Movies dataset, but it does not include multicriteria ratings.[4] To address this issue, the authors of [P3] created a multicriteria version of the same dataset by crawling the Yahoo! Movies website. In total, 36 studies exploit the Yahoo! Movies dataset, either the version obtained by Jannach et al. [P3] or other versions crawled by different researchers, such as [P18], [P29], and [P53].

TripAdvisor

TripAdvisor is a website that contains restaurant and hotel reviews. Similarly to Yahoo! Movies, there is no official multicriteria rating dataset, as different researchers crawled the website and created their own version, typically exploiting hotel ratings and reviews. For example, this approach was followed by the authors of [P14] and [P83]. The TripAdvisor dataset collected by the authors of [P1] is publicly available.[5] Also the TripAdvisor dataset created by Wang et al. [133] and

[4] https://webscope.sandbox.yahoo.com

[5] https://www.cs.cmu.edu/~jiweil/html/hotel-review.html

used in [P91] is available online.[6] In total, the TripAdvisor dataset was mentioned by 19 studies.

In-house

We identified 9 studies that created a multicriteria dataset in-house for conducting an offline comparison. For example, the authors of [P60] collected 9,628 ratings about songs that were later used to evaluate a music recommender system. In [P19], different students were invited to provide ratings about universities.

MovieLens

Even if the MovieLens datasets only contain single criteria ratings, they were also exploited for evaluating multicriteria recommender systems. For instance, in [P2] and in [P61], MovieLens 100K was transformed in a four criteria dataset. A similar approach was followed in [P48] with MovieLens 10M, where a method capable of extracting multicriteria preferences from traditional ratings using external aggregate ratings and descriptive data is discussed. In total, the MovieLens datasets were mentioned by 5 studies.

Synthetic

Because of the lack of public multicriteria datasets, some researchers created synthetic ratings in order to evaluate their approach. For example, the authors of [P78] simulated a dataset about equity fund recommendations. In [P77], a testing tool named CollaFiS, capable of building multicriteria datasets, is discussed. This approach was followed by 4 studies.

Other Datasets

Less common datasets, exploited by 1 or 2 studies each, include, for example, Bcn Restaurantes [P39], HRS.com [P91], and RateBeer [P93]. Such various datasets were considered by 18 studies.

3.2.9 Future Works

In the following, we analyze the suggestions for future works mentioned in the reviewed studies in order to provide an answer to RQ1.9. A summary of our findings is available in Table 3.10.

[6] http://www.cs.virginia.edu/~hw5x/Data/LARA/TripAdvisor

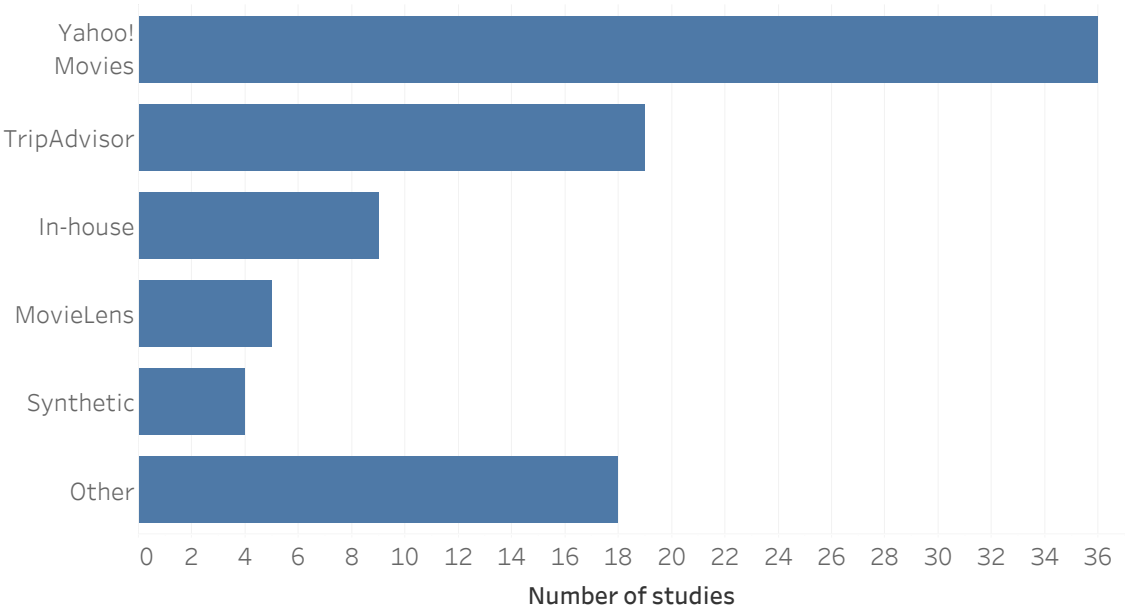


Figure 3.7: This barplot represents the number of studies per evaluation dataset.

Extend the Solution

Different authors propose to extend or modify the described recommender system for increasing its accuracy. This future work is related to the problem of personalization, discussed in Section 3.2.2. Common suggestions include adding additional components like clustering algorithms or further recommendation models and exploiting soft-computing techniques. In total, we identified this future work in 33 studies.

Include More Data or Additional Criteria

Another possibility is to improve the proposed approach by including more data or by increasing the number of criteria exploited by the recommendation algorithm. For example, it is possible to rely on external ontologies, contextual and content information, trust-related scores, and also consider additional criteria extracted from user reviews. This category of future works was mentioned in 24 studies.

Improve the Evaluation

Some studies mention the fact that the evaluation performed by their authors was not enough complete or detailed because of different kinds of constrains. For this reason, it would be advisable to increase its trustfulness by executing it again considering more datasets, techniques, and evaluation metrics. In total, this problem was reported by 14 studies.

Identify Significant Criteria

A common issue associated with multicriteria recommenders is the noise introduced by redundant criteria, as discussed in Section 3.2.2. For this reason, some authors considered the identification of the most significant criteria as a future work. This suggestion was discussed in 13 studies.

Increase the Scalability

A typical issue of multicriteria recommender systems is their limited scalability. This problem was highlighted by different studies in Section 3.2.2. Some researchers suggested to study how to increase it, for example by means of parallel computational paradigms. We identified this future work in 10 studies.

Consider Different Domains

Finally, 7 studies mentioned the necessity of validating the proposed approach in different domains, like it is usually done with traditional recommenders. However, this objective is difficult to achieve because of the limited availability of multicriteria rating datasets.

Other Future Works

Further categories of future works, mentioned in less than 5 studies each, include designing solutions for addressing the cold-start problem, as described in Section 3.2.2, performing experiments with adaptive recommenders, solving the issues related to preference elicitation, creating algorithms for explaining the recommendations, and performing an analysis of the related ethical problems.

Future work	Studies
Extend the solution	33
Include more data or criteria	24
Improve the evaluation	14
Identify significant criteria	13
Increase the scalability	10
Consider different domains	7
Other	9

Table 3.10: The number of studies per category of future work.

3.3 Discussion

In the following, we discuss the outcome of our systematic literature review, considering the answers provided in Section 3.2 to the research questions originally introduced in Section 3.1.1, and highlighting possible threats to validity.

3.3.1 Included Studies

As reported in Figure 3.2, the earliest study included in this review, which is [P76], dates back to the year 2003. However, the field of multicriteria recommender systems started to be relatively widespread only from the year 2007, when influential studies like [P53] were published. We can also observe an increasing amount of publications, suggesting that this research topic is still popular, as recommender systems in general. In particular, the last few years were characterized by a higher number of studies related to specialized applications of multicriteria approaches, for example in tourism, health and care, and distance learning.

Regarding the quality of the included studies, summarized in Figure 3.3, we can highlight the fact that higher scores were assigned to works published in journals with respect to conferences. This result is consistent with the conclusions of other systematic literature reviews conducted in related fields, like hybrid and linked data-based recommender systems [21, 44].

3.3.2 Research Problems

By looking at the problems listed in Section 3.2.2, it is possible to observe that the most frequent one faced by researchers is data sparsity. This is a general issue of collaborative filtering recommender systems that causes a lower recommendation quality due to an insufficient amount of input data. However, it is reasonable to suppose that data sparsity is more severe in the context of multicriteria recommenders, because of the higher amount of expected ratings. Possible solutions include the use of external information or the construction of latent models. In contrast, it is not clear what is the effect of using different rating elicitation methods.

Another typical research problem is discovering what are the optimal weights for each criterion. They may be computed globally or for each user with the objective of maximizing the recommendation accuracy. A second approach is to obtain the weights directly from the user, for example with decision making techniques.

A different but related issue is represented by data noise, caused by redundant criteria or dishonest ratings. Of course, in order to minimize the user's effort and the probability of obtaining inaccurate ratings, it is necessary to limit the number of criteria. However, identifying the most appropriate ones for a given domain is a difficult task even for an expert. In our opinion, this issue should be better investigated by future studies.

Accuracy is a characteristic required for any machine learning technique and, in the context of recommender systems, it is related to user satisfaction. Multicriteria recommenders can increase the personalization of the suggestions if they are capable of correctly identifying what are the most important criteria for each user. However, when a user is new to the system, the cold-start problem arises. This is a general issue of collaborative filtering recommenders and it is typically solved by creating hybrid solutions that consider content-based information.

Finally, the usage of multiple criterion results in algorithms that are less scalable. This problem can be addressed with clustering and dimensionality reduction techniques, as well as by exploiting distributed architectures.

3.3.3 Recommendation Approaches

The vast majority of multicriteria recommender systems can be classified as collaborative filtering approaches, as reported in Table 3.6. For example, in heuristic-based methods, the ratings provided for each dimension are exploited together using a multidimensional distance metric, extending the traditional neighborhood-based recommendation technique. More complex heuristics rely on the aggregation of different similarities computed per criterion, possibly using weights specific for each user. Also in the context of model-based collaborative filtering, methods like matrix factorization are applied to each dimension and, then, their results are aggregated in a global predicted rating.

For this reason, almost all recommender systems included in this review can be considered hybrid, as they combine multiple collaborative filtering models, one for each criterion. Furthermore, such techniques are sometimes exploited together with content-based, knowledge-based, community-based, and demographic approaches, resulting in other forms of hybrid recommenders [19].

However, a limited number of studies included in this review deals with recommender systems that can be classified only as knowledge-based. Such systems exploit domain specific knowledge to match the available items with the user preferences. They were selected because we identified them as a form of multicriteria recommendation, even if the distinction between knowledge-based recommenders and traditional information retrieval methods is not always clear.

3.3.4 Multicriteria Techniques

In Section 3.2.4, we reported that the most common technique exploited by multicriteria recommenders is the k-NN algorithm. This result is consistent with the recommendation approaches identified in the reviewed studies, as neighborhood-based methods represent a popular approach of collaborative filtering. k-NN may be enough to build a multicriteria recommender: the similarities among users or items can be directly computed with a multidimensional distance metric and they

can be aggregated using a trivial approach, like the averaging function, or estimated by means of more complex techniques.

In contrast, multicriteria matrix manipulation methods represent a less common approach to collaborative filtering and they are often exploited to estimate unknown ratings for each criterion or to reduce the dimensionality of the problem. Again, matrix manipulation methods are usually combined with other methods designed to estimate importance weights for each criterion.

Different families of such techniques have been identified in the reviewed studies namely, regression analysis, neural networks, and genetic algorithms. It is possible to produce an overall rating using regression analysis if the other methods are not capable of dealing with multiple ratings on their own. A more recent alternative to regression analysis is represented by neural networks, that are exploited for learning the relative importance of criteria. Finally, genetic algorithms may be used to discover the weights of each criterion, but also to estimate good parameters for the recommendation model.

For reducing the complexity of the problem, some studies considered clustering algorithms in order to create cohesive groups of similar users or items. Such algorithms are usually applied as a first step, before other techniques capable of creating suggestions suitable for each cluster. In contrast, fuzzy logic may be exploited together with all the aforementioned methods for formally encoding the fact that ratings usually represent uncertain values.

The second most popular recommendation method after k-NN consists in decision analysis techniques. They are typically used to generate an ordered list of suggested items considering potential conflicts in user requirements. Decision analysis algorithms are often combined with k-NN, matrix manipulation, and clustering methods. However, some studies do not mention a specific recommendation technique to be exploited jointly with a decision analysis method because the proposed approach is presented as a general framework and, thus, any recommendation method is suitable.

In Table 3.11, we report the main benefits and issues of the reviewed multicriteria methods. They could be considered as different compromises between the correctness of the suggestions and the complexity of the approach. Thus, it is of paramount importance being able to decide if, for a given task, it is more appropriate to foster the scalability of the system or the accuracy of the results.

Finally, in Figure 3.8, we summarize how such techniques could be exploited for constructing an ideal multicriteria recommender. In the preprocessing phase, common approaches designed to reduce the dimension of the input ratings are clustering and genetic algorithms. Also fuzzy logic may be applied at this point to transform the ratings in fuzzy numbers. During the rating prediction phase, it is possible to rely on k-NN, matrix manipulation methods, neural networks, and statistical models. These algorithms may already include a dimensionality reduction step and a way of obtaining an ordered list of suggestions. Finally, in

Technique	Benefit	Issue
k-NN	It can compute directly the predictions using a multidimensional metric.	If it is combined with other methods its scalability is reduced.
Decision analysis	It can analyze the importance that each user gives to the available criteria.	It may require additional data from the users apart their multicriteria ratings.
Fuzzy logic	It supposes that rating values may have a different meaning depending on the user.	It must be combined with additional techniques to actually recommend items.
Regression analysis	It is the most popular approach for discovering the individual weight of each criteria.	It must be combined with k-NN or matrix manipulation.
Clustering	It can create groups of users with similar rating behaviours and reduce the dimension of the problem inputs.	It must be exploited together with additional techniques.
Matrix manipulation	It can compute unknown ratings for each criteria in an efficient and effective way.	It usually require a final step to calculate an overall predicted rating.
Neural network	Recent approaches can directly predict the final ratings with an interesting accuracy.	It is often exploited to compute the weights of each criteria, adding more complexity.
Generic algorithm	It can automatically determine the optimal weight for each criteria and user.	It must be combined with additional techniques to predict the final ratings.

Table 3.11: The main benefits and issues of the reviewed techniques.

the ranking phase, popular approaches are decision analysis, regression analysis, neural networks, and genetic algorithms. Please note that the first and third phase are, in general, optional and they add several layers of complexity to the developed system. Furthermore, some techniques may already perform these tasks during the rating prediction phase.

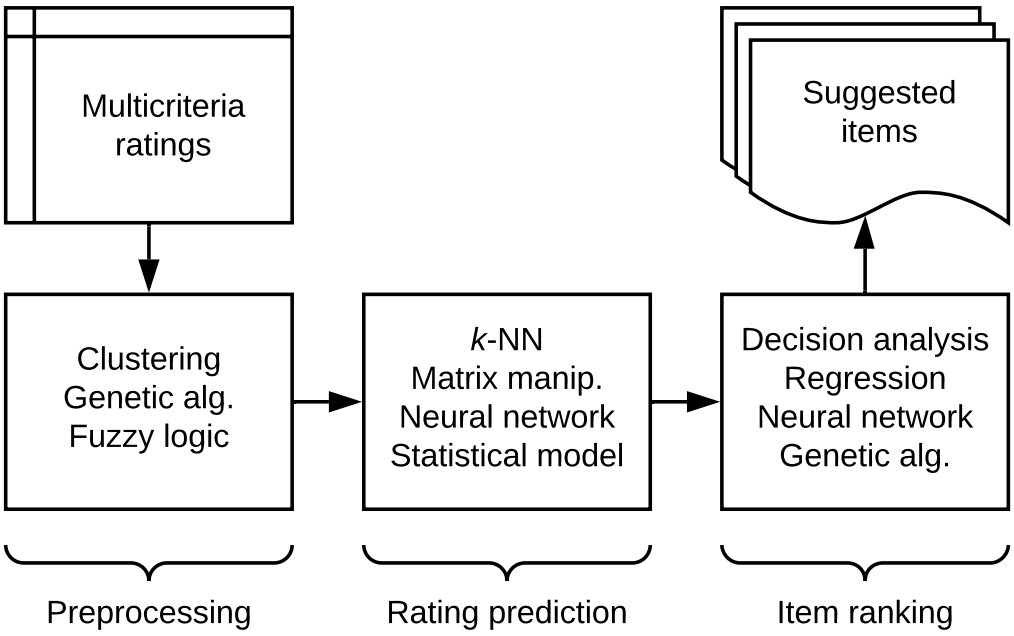


Figure 3.8: This diagram represents the possible techniques exploited by an ideal multicriteria recommender, subdivided according to the recommendation phase. Content-based features and other less common approaches are not considered.

3.3.5 Application Domains

As already observed in Section 3.2.5, multicriteria recommenders are frequently exploited in the tourism and travel domain. This result is expected, as many travel websites, like TripAdvisor[7] and Bcn Restaurantes,[8] rely on multicriteria ratings for suggesting items to their users. Even for someone that is not a domain expert, it is possible to identify some useful criteria that should be considered when evaluating a hotel or a restaurant.

For this reason, we can suppose that items belonging to complex domains can be more easily analyzed considering different dimensions and, therefore, they are

[7] https://www.tripadvisor.com

[8] https://www.bcnrestaurantes.com

better suited for multicriteria recommender systems. Another popular example is represented by consumer products, which can be evaluated considering different criteria based on their features, price, and quality.

On the other hand, domains like music or books are usually not exploited for creating a multicriteria recommender, probably because assigning detailed evaluations of such items is too difficult for someone that is not a domain expert.

Another factor that must be considered before performing further analysis is the availability of a certain multicriteria rating dataset in a particular domain. For example, many researchers used to collected ratings from the Yahoo! Movies website, but nowadays such service is no longer available.

The majority of the recommender systems presented in the reviewed studies are domain-independent, as they are only based on multicriteria ratings. However, some solutions designed for specific domains, like medical treatments and research papers, cannot be easily adapted to other scenarios.

3.3.6 Evaluation Protocols

In general, performing the evaluation of a recommender system is a complex task, because the slightest change in the experimental protocol may produce radically different results [116]. It is not possible to directly compare the outcomes of different works because of the great variety of datasets, evaluation protocols, and metrics mentioned in the reviewed studies.

The multicriteria recommender systems were often evaluated using offline comparisons, as summarized in Table 3.8. This result is expected, as such an approach is less expensive than performing a user study, but it can be used to obtain useful information. However, in order to report more conclusive results, some authors decided to perform a user study. We noticed that it is difficult to involve a significant number of experimental subjects. In fact, only in 3 studies the number of participants was higher than a hundred people.

3.3.7 Evaluation Metrics

The majority of the reviewed studies validated their approach using rating accuracy metrics, as reported in Table 3.9. This result is probably related to the fact that recommender systems were initially considered as tools designed to predict unknown ratings as accurately as possible. This evaluation approach produces results that are interesting from an academic point of view, but they may have a limited practical meaning.

In fact, as suggested in [61], it is not possible to display in an application all the items that are associated with the highest predicted ratings. Therefore, it is necessary to be capable of correctly creating lists of ranked items that will be shown

to users. This aspect is considered by different studies, in which usage and ranking accuracy metrics are reported.

Other categories of metrics identified during the review include coverage and scalability, which were evaluated in a relatively low number of studies. It is interesting to notice that more recent evaluation dimensions, like diversity, novelty and serendipity, have not been considered, suggesting that their adoption in the recommender system community is still limited.

Finally, we also observed that the differences in the evaluation protocols followed by the reviewed studies do not allow a direct comparison of the results, even when widespread metrics, like MAE or precision, are exploited. For example, we identified a wide variability in the rating dataset, the splitting method between training and test set, and the definition of non-relevant items. This conclusion is consistent with the findings reported in [116].

3.3.8 Evaluation Datasets

From the results presented in Section 3.2.8, we observe that different evaluation datasets have been created using data available in online portals that contain multicriteria ratings. Such collections of ratings are usually not directly provided by the platforms, like the MovieLens datasets, but they are downloaded by researchers using web crawling methods. These approaches result in different versions of the same dataset that sometimes are not publicly available and they can only be obtained by contacting the original creators.

Consider, for example, Yahoo! Movies: this dataset was realized by adding multicriteria ratings to a collection of movie preferences initially provided by Yahoo! Research, quickly becoming one of the most widespread multicriteria gold standards. However, the Yahoo! Movies platform does not provide multicriteria ratings anymore, making impossible to recreate it. Similar observations are also valid for other datasets, as platforms may change over the years and original datasets may become unavailable. Anyway, we were able to locate two TripAdvisor dataset versions that are still publicly available on the Web.

It is interesting to notice that the authors of different studies decided to create new datasets, relying on real users and simulation tools, or to add additional criteria to existing datasets. This fact suggests that there is the need of realizing and publishing further multicriteria datasets.

3.3.9 Future Works

We described the main future work directions reported by the authors of the reviewed studies in Section 3.2.9, in order to identify how it is possible to advance the multicriteria recommender field. The most common suggestion is to add to the proposed technique an additional component for improving its personalization.

The authors of such recommenders usually observe that their methods could benefit from further preprocessing phases or additional components to be combined in a hybrid approach. Another possibility is to consider soft-computing techniques for expressing the uncertainty and vagueness of user ratings.

It is interesting to notice that some studies propose to improve multicriteria recommenders by considering further data, while other authors are more interested in reducing the number of available criteria. These suggestions are not in contrast, as they may be related to the exploited dataset. Therefore, correctly defining the rating criteria and better studying how such ratings are collected are of paramount importance for obtaining high quality recommendations.

Different works propose to also consider friendship relations among users, external knowledge bases and ontologies, and more contextual information. Social Recommender Systems (SRS) [56], Semantics-Aware Recommender Systems (SARS) [49], and Context-Aware Recommender Systems (CARS) [3] are relatively new recommendation approaches that could be exploited jointly with multicriteria techniques in order to create a more personalized experience. However, such additional data need to be managed in a proper way: for this reason, some researchers propose to also study the scalability of the available solutions.

As already discussed, recommender system evaluation is not an easy task. Therefore, different authors declare that they plan to conduct a more extensive experimentation of the proposed approach in future works. This frequent situation may be caused by the difficulties in obtaining multicriteria datasets or by the elevated costs necessary for involving real users in the evaluation process. A related future work is to evaluate the same approach in different domains, as also this suggestion requires further datasets and experiments.

3.3.10 Threats to Validity

In this section, we point out the main problems that we addressed while conducting this systematic literature review. We noticed that there is no strong agreement on a shared definition of multicriteria recommender systems, as there are some subtle differences but also some similarities with multiattribute content-filtering techniques [2]. For this reason, we only considered as multicriteria the recommenders based on multicriteria ratings. We also included in our selection few studies describing methods that cannot be considered collaborative filtering approaches, but that exploit multicriteria ratings in a non-conventional way. Furthermore, a limited number of studies were initially selected by analyzing their title and abstract. However, we were later forced to exclude them even if they were potentially relevant because we were not able to retrieve their full-text version. We also noticed that some studies were described in different research papers, and, therefore, we analyzed these situations with attention. Sometimes we selected multiple works because we discovered that these contributions were only partially overlapped.

3.4 Conclusion

In the context of this systematic literature review, we analyzed a total number of 93 studies related to the topic of multicriteria recommender systems. We provided an answer to nine different research questions, in order to identify which are the main problems that multicriteria recommenders aim to resolve, what recommendation approaches they usually exploit, and what are the most common machine learning and data mining techniques considered for realizing them. Furthermore, we investigated in which domains multicriteria recommenders can be applied, how they are evaluated in terms of experimental protocols, metrics, and rating datasets. Finally, we reported the most common suggestions available in selected studies for conducting future research works.

We identified data sparsity as the most frequent problem that researchers try to address. This issue could be caused by the fact that users are less willing to provide multicriteria ratings because they find this task difficult and time consuming, as they need to consider different aspects of the same item separately. A related research problem is understating what are the most appropriate criteria that should be considered in a particular domain. An excessive number of criteria will most likely result in a partial overlap of their meanings, thus causing a reduction of the prediction accuracy because of the additional noise in the input data. Further studies should better quantify what is the impact of this problem and how recommendation algorithms could reduce it.

We observed that almost all multicriteria recommenders rely on collaborative filtering techniques. This approach is, in fact, also popular in the context of single-criteria recommenders. A large portion of the reviewed studies propose hybrid recommendation methodologies, as they usually combine multiple runs of collaborative filtering algorithms, one for each criterion. However, we also observed a more recent trend of also relying on additional techniques, like content-based, knowledge-based, and community-based approaches. The motivating idea of such recommender systems is to increase the accuracy of the suggested items by exploiting further data behind pure multicriteria ratings.

Furthermore, we classified the machine learning and data mining techniques mentioned in the reviewed studies according to the recommendation phase in which they are considered. We observed that during the preprocessing phase different authors choose to reduce the dimension of the problem inputs by means of clustering and genetic algorithms. In the rating prediction phase, various collaborative filtering methods are considered, like k-NN, matrix manipulation techniques, neural networks, and statistical models. Finally, during the ranking phase, it is possible to rely on decision analysis, regression analysis, neural networks, and genetic algorithms. Such methods are combined in various ways, and sometimes integrated with the results of additional recommendation approaches.

We also investigated the domains that are usually considered for implementing a

multicriteria recommender system. In the reviewed studies we frequently observed items related to the tourism and travel domain, for example hotels and restaurants. Another popular domain is represented by movies, probably due to the past availability of the Yahoo! Movies datasets. In contrast, domains that are often exploited in single-criteria recommenders, like music or books, are almost never considered for multicriteria approaches. Possible explanations of this finding could be the lack of proper evaluation datasets or the difficulties in defining meaningful criteria and obtaining the associated ratings.

From the analyzed experiments we observed that it is not possible to directly compare the results obtained with different recommendation approaches due to the extreme variability in the experimental protocols followed by their authors. Another critical point that emerged from this work is the lack of publicly available multicriteria rating datasets. Even if this problem could be partially mitigated with synthetic and crawled datasets, additional efforts should be done in order to create proper benchmarks in different domains and, thus, enabling a better reproducibility of the experimental results.

Finally, our findings indicate that future works in this field should better explore how to increase the utility of multicriteria recommenders by integrating community-based, knowledge-based, and context-aware approaches, according to the peculiar characteristics of the recommended items. For example, a restaurant recommender system would greatly benefit from context-aware information, a movie recommender could also leverage on semantic data, while a consumer product recommender may assign more importance to the opinions of trusted users. Another interesting point that should be better addressed in future works is defining what are the most appropriate criteria for each domain and how the ratings associated to that criteria should be collected in order to minimize user efforts, also considering soft-computing and linguistic techniques. Furthermore, it would be useful to explore novel algorithms for dynamically understanding the importance that each user implicitly assigns to a criterion, for reducing the fatigue associated with the preference elicitation phase.

Chapter 4

Evaluation of Sequence-based Recommender Systems

From the outcome of our systematic literature review on multicriteria recommender systems discussed in Chapter 3, it emerged that traditional approaches usually guarantee interesting results in well-known domains, such as movie recommendation, but they are not capable of capturing the temporal evolution of users' preferences [20]. However, different authors [40, 110, 59] argue that movies watched recently provide more useful information about a certain user than those she consumed in a distant past. It is, in fact, reasonable to assume that a recent item may have a high influence on the choice of the next one.

A recommender system that exploits sequential data for predicting the sole next item that will be consumed by a user can be defined as *sequential recommender* [134]. Several works related to sequential recommenders are available in literature, as discussed in Section 2.1.2. For example, Zhou et al. [140] exploited a sequential pattern mining algorithm for recommending which page to visit next in a website, while Rendle et al. [110] relied on Markov chains for suggesting products considering previous purchases. Recently, He et al. [59] designed a recommender system capable of modeling how users' interests evolve over time.

While all these methods usually consider user preferences observed during the training phase as sequences, no temporal ordering is available at recommendation time, as only one item, or a list of items ranked by relevance, is suggested to users. Because of the popularity of CF techniques, most of sequential recommenders are based on such approaches [108], but in principle it is also possible to design systems capable of analyzing sequences according to content-based methods.

In general, even if the problem of creating a sequence of words starting from an initial one is a well-known task inside the natural language processing community [73], the idea of creating personalized sequences of items is less widespread in the context of recommender systems [62]. For this reason, it would be interesting to be able to exploit the temporal ordering not only during the training phase but

also for generating sequences of recommended items, such as in the task of language modeling. Some solutions to this problem have already been proposed in industry, and also few researchers have discussed how to automatically construct music playlists [28] or suggest sequences of points-of-interest to tourists [43] starting from seed items. However, early studies conducted in this field lack of a common definition of the problem that they are trying to address. For example, *session-based* recommenders only consider the last session of the current user [82], while *sequence-aware* ones also exploit the history of past sessions [108] and they can be considered equivalent to sequential recommenders. Furthermore, it is not clear if item repetitions are allowed in the suggestions or not.

In this chapter, we argue that it is possible to consider recommender systems capable of creating personalized sequences of an arbitrary length as a generalization of a sequential recommender because the latter is only able of creating sequences of length *one*. In contrast to traditional approaches that usually create lists of items ranked by relevance, in the following we will define a recommender that exploits a temporal dimension both in the training and in the generation phase as a *sequence-based* recommender, as it observes and suggests sequences of items meant to be consumed in a particular order.

As discussed in Section 2.2, several evaluation protocols and metrics for analyzing novel recommenders via offline experiments are available, to capture the different aspects of the recommendation algorithm [55]. However, the lack of a standardized way of performing such *in vitro* experiments leads to results that are often incomparable [70]. To the best of our knowledge, no evaluation framework for sequence-based recommender systems has already been proposed. Our motivating hypothesis is that, in the context of sequence-based approaches, traditional evaluation metrics need to be computed at the level of sequences instead of the level of users. Therefore, we shift our focus from the multicriteria nature of the input ratings to multicriteria evaluation approaches, that rely on the set of metrics to perform an offline comparison among different recommendation methods.

In our view, an evaluation framework consists of a methodology for performing an experimental comparison, a set of metrics, and a software tool that implements them. The main aim of this chapter is to address the following research questions and to introduce an offline evaluation framework for sequence-based recommender systems that we called Sequeval.

RQ2.1 What is the formal definition of a sequence-based recommender system?

RQ2.2 How already established metrics can be extended and adapted for evaluating a sequence-based recommender system?

RQ2.3 Against which baseline approaches a sequence-based recommender system can be compared?

Because our evaluation approach is agnostic with respect to the implementation details of the algorithms under analysis, it can be successfully exploited to also assess the performance of systems based on alternative recommendation methods [32] or dealing with unconventional categories of items, for example 3D movies [106] or cultural digital contents [9], especially if they are supposed to be consumed by users in a sequential order. Besides, by openly releasing a Python implementation of Sequeval, we aim to encourage the use of the proposed framework as an attempt to standardize the evaluation of sequence-based recommender systems, mitigating the comparability problem in recommendation research.

The remainder of this chapter is organized as follows: in Section 4.1 we present the mathematical definition of a sequence-based recommender system; in Section 4.2 we introduce Sequeval by describing its evaluation protocol, metrics, and implementation details; in Section 4.3 we perform an empirical analysis of the framework with two datasets. Finally, in Section 4.4, we formulate our conclusions.

4.1 Sequence-based Recommender Systems

Before analyzing the evaluation framework, we introduce the problem of recommending sequences and we provide an answer to RQ2.1. In a traditional recommender system, users express positive or negative preferences about a certain item. An item may be, for example, a product, a song, or a place. In contrast, we assume that when a user consumes or interacts with an item, she expresses an implicit rating about it. This assumption in the literature goes under the name of *implicit feedback* [62]. Because we are also considering the temporal dimension to build the sequences, each rating is associated with a timestamp that represents the point in time when it was recorded.

Definition 1. *Given the space of items $\mathcal{I}$, the space of users $\mathcal{U}$, the space of timestamps $\mathcal{T}$, a rating $\mathbf{r} \in \mathcal{R}$ is a tuple $\mathbf{r} = (\iota, \upsilon, \tau)$, where $\iota \in \mathcal{I}$ is the item for which the user $\upsilon \in \mathcal{U}$ expressed a positive preference at the timestamp $\tau \in \mathcal{T}$.*

By relying on the set of ratings $\mathcal{R}$ available in the system, it is possible to construct the sequences that will be used to train and to evaluate the recommender. Each sequence only includes the ratings expressed by a single user. On the other hand, each user may produce several sequences.

The concept of *sequence* is similar to the concept of *session* in a traditional web interaction: if two ratings are distant in time more than an interval $\delta\tau$, then they belong to different sequences. Some ratings may be isolated and, for this reason, not part of any sequence. The most appropriate value for $\delta\tau$ depends on the domain: for example, in the POI recommendation scenario, it could be considered of a few hours as reported in [43].

Definition 2. *A sequence* $\mathbf{s} \in \mathcal{S}$ *is a temporally ordered list of ratings* $\langle \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n \rangle$ *created by a particular user* $\upsilon \in \mathcal{U}$*, i.e., for each* i*,* $\mathbf{r}_i = (\iota_i, \upsilon, \tau_i)$ *and* $\tau_i < \tau_{i+1}$*.*

In Algorithm 4.1, we list the procedure for creating the set $\mathcal{S}$, given the set of users $\mathcal{U}$, the set of ratings $\mathcal{R}$, and a time interval $\delta\tau$. Please note that we do not allow the creation of sequences of length *one* because they do not encode a meaningful temporal order.

Algorithm 4.1 Generation of the set $\mathcal{S}$, given $\mathcal{U}$, $\mathcal{R}$, and $\delta\tau$.

Require: $\mathcal{U} \neq \{\emptyset\} \wedge \mathcal{R} \neq \{\emptyset\} \wedge \delta\tau \doteq \tau_i - \tau_j$

$\mathcal{S} \leftarrow \{\emptyset\}$
for all $\upsilon \in \mathcal{U}$ **do**
  $\mathbf{s} \leftarrow \varnothing$
  for all $\mathbf{r}_i \in \mathcal{R}_\upsilon : \tau_{i-1} < \tau_i \wedge i > 1$ **do**
    if $\tau_i < \tau_{i-1} + \delta\tau$ **then**
      if $\mathbf{s}$ is $\varnothing$ **then**
        $\mathbf{s} \leftarrow \langle \mathbf{r}_{i-1} \rangle$
      end if
      $\mathbf{s} \leftarrow \mathbf{s} + \langle \mathbf{r}_i \rangle$
    else
      if $|\mathcal{R}_\mathbf{s}| > 1$ **then**
        $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{s}\}$
        $\mathbf{s} \leftarrow \varnothing$
      end if
    end if
  end for
end for
return $\mathcal{S}$

A sequence-based recommender is a recommender system capable of suggesting a personalized sequence that is built starting from a seed rating $\mathbf{r}_0$, considering the example sequences already available in the system and the specific behavior of a certain user. The seed rating is characterized by a seed item ι_0, a target user υ, and an initial timestamp τ_0. The seed item can be represented by any item that belongs to the catalog, but, more in general, it is a point in the space of items $\mathcal{I}$. For example, in the music domain, it could identify not only a particular song, but also an artist, a genre, or a mood. The target user is the user to whom the sequence is recommended, while the initial timestamp represents the point in time in which the recommendation is created. The generated sequence is of a fixed length and it contains exactly k ratings. Please note that if $k = 1$, we are dealing with a sequential recommender as defined in [134].

Definition 3. *Given a seed rating* $\mathbf{r}_0 \in \mathcal{R} : \mathbf{r}_0 = (\iota_0, \upsilon, \tau_0)$*, and a length* $k \in$

$\mathbb{N}$, *a sequence-based recommender is the function* sequence : $\mathcal{R} \times \mathbb{N} \rightarrow \mathcal{S}$, *i.e.,* sequence$(\mathbf{r}_0, k) = \langle \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_k \rangle : \mathbf{r}_i = (\iota_i, \upsilon, \tau_i)$.

Most sequence-based recommenders are based on probability models, and therefore they can be interpreted as a sampling function σ applied to the conditional probability $P(\langle \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_k \rangle | \mathbf{r}_0)$:

$$\text{sequence}(\mathbf{r}_0, k) = \sigma(P(\langle \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_k \rangle | \mathbf{r}_0)) \tag{4.1}$$

Using the chain rule, the sequence probability $P(\langle \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_k \rangle | \mathbf{r}_0)$ can be written as:

$$\begin{aligned} P(\langle \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_k \rangle | \mathbf{r}_0) = & P(\mathbf{r}_k | \langle \mathbf{r}_0, \ldots, \mathbf{r}_{k-1} \rangle) \cdot \\ & P(\mathbf{r}_{k-1} | \langle \mathbf{r}_0, \ldots, \mathbf{r}_{k-2} \rangle) \cdots P(\mathbf{r}_1 | \langle \mathbf{r}_0 \rangle) \end{aligned} \tag{4.2}$$

For example, in the case of a Markov chain, each rating depends on the previous one, i.e., $P(\mathbf{r}_k | \langle \mathbf{r}_0, \ldots, \mathbf{r}_{k-1} \rangle) = P(\mathbf{r}_k | \mathbf{r}_{k-1})$:

$$P(\langle \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_k \rangle | \mathbf{r}_0) = P(\mathbf{r}_k | \mathbf{r}_{k-1}) P(\mathbf{r}_{k-1} | \mathbf{r}_{k-2}) \cdots P(\mathbf{r}_1 | \mathbf{r}_0) \tag{4.3}$$

Thus, a sequence-based recommender system typically works by learning from a set of sequences $\mathcal{S}_{training}$ the conditional probability of the next rating $\mathbf{r}_k$ to the sequence of previous ones $\langle \mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{k-1} \rangle$, i.e., the factors of the right-hand side of Equation 4.2. Sampling sequences directly from Equation 4.2 would require computing the probabilities of all the $|I|^k$ possible sequences, where $|I|$ is the size of the vocabulary of items and k is the length of the sequences. Since this becomes easily computationally unfeasible, we opt for a greedy approach, in which at each step we sample the next most likely item. A sampling function ρ is defined to select a particular next rating from the previous ones at each step:

$$\hat{\mathbf{r}}_k = \rho(P(\mathbf{r}_k | \langle \mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{k-1} \rangle)) \tag{4.4}$$

A trivial example of ρ is the *argmax* function, which simply selects the most probable next rating. In the following, we will assume that ρ is implemented by a weighted random sampling function.

Algorithm 4.2 formalizes the procedure for generating a personalized sequence, given a seed rating $\mathbf{r}_0$, and a length k, i.e., it describes the sampling function σ. For k times, the next rating of the recommended sequence is generated using the function *predict*. The function *predict* : $\mathcal{S} \rightarrow \mathcal{R}$ implements the sampling function ρ and it returns the most probable next rating for the current input sequence. In practice, the sequence-based recommender system can estimate the probability that the next rating of the current sequence will include a particular item at a certain timestamp.

Please note that the greedy procedure described in Algorithm 4.2 is not the only way to create samples of sequences, and thus, the user of the evaluation framework or the designer of the recommendation method are free to define other ways and strategies for that end.

Algorithm 4.2 Recommendation of a sequence of length k.

Require: $\mathbf{r}_0 \in \mathcal{R} \wedge k > 0$
1: $\mathbf{s} \leftarrow \langle \mathbf{r}_0 \rangle$
2: **for** $i = 1$ **to** k **do**
3: $\quad \mathbf{r}_i \leftarrow \text{predict}(\mathbf{s})$
4: $\quad \mathbf{s} \leftarrow \mathbf{s} + \langle \mathbf{r}_i \rangle$
5: **end for**
6: **return** $\mathbf{s} - \langle \mathbf{r}_0 \rangle$

To compute some metrics that are part of the evaluation framework, it is necessary to know the number of items that are associated with a certain sequence. For this reason, we define the set $\mathcal{I}_\mathbf{s}$ as the set of items that are part of the sequence $\mathbf{s}$, and the set $\mathcal{R}_\mathbf{s}$ as the set of ratings that are part of the sequence $\mathbf{s}$. Therefore, $|\mathcal{I}_\mathbf{s}|$ is the number of distinct items available in $\mathbf{s}$, while $|\mathcal{R}_\mathbf{s}|$ represents the length of $\mathbf{s}$, i.e., the number of ratings available in $\mathbf{s}$.

For instance, we can suppose that the set of ratings $\mathcal{R}$ is equal to $\{(\iota_1, \upsilon_1, \tau_1), (\iota_2, \upsilon_1, \tau_2), (\iota_3, \upsilon_2, \tau_3), (\iota_1, \upsilon_1, \tau_4), (\iota_2, \upsilon_2, \tau_5), (\iota_3, \upsilon_1, \tau_6)\}$. Then, if we assume that the only pair of timestamps that violates the $\delta\tau$ constraint is (τ_4, τ_6), we can create two sequences: $\mathbf{s}_1 = \langle (\iota_1, \upsilon_1, \tau_1), (\iota_2, \upsilon_1, \tau_2), (\iota_1, \upsilon_1, \tau_4) \rangle$, and $\mathbf{s}_2 = \langle (\iota_3, \upsilon_2, \tau_3), (\iota_2, \upsilon_2, \tau_5) \rangle$. The rating $(\iota_3, \upsilon_1, \tau_6)$ is not part of any sequence because it was created at some point in time later than $\tau_4 + \delta\tau$ and we do not have any subsequent rating expressed by υ_1. We also observe that $|\mathcal{I}_{\mathbf{s}_1}| = 2$ and $|\mathcal{R}_{\mathbf{s}_1}| = 3$. We would like to recommend a sequence $\mathbf{s}$ of length *two* to user υ_1 starting from item ι_3 at timestamp $\tau_{\mathbf{s},0}$. In fact, it is not required that the item ι_3 already appeared in the sequences related to user υ_1. A possible solution to this problem is to define $\mathbf{r}_{\mathbf{s},0} = (\iota_3, \upsilon_1, \tau_{\mathbf{s},0})$ and then to recommend $\mathbf{s} = \text{sequence}(\mathbf{r}_{\mathbf{s},0}, 2) = \langle (\iota_2, \upsilon_1, \tau_{\mathbf{s},1}), (\iota_1, \upsilon_1, \tau_{\mathbf{s},2}) \rangle$, where $\tau_{\mathbf{s},1}$ and $\tau_{\mathbf{s},2}$ may be used to suggest when consuming the items.

4.2 Sequeval Evaluation Framework

Comparing the performance of several recommenders with an experiment that involves a live system is not always feasible or appropriate. For this reason, it is necessary to first perform a preliminary evaluation in an offline scenario [55]. In such a setting, we can assess the performance of the algorithms by comparing with baselines, understanding the strengths and weaknesses of those, and thus avoiding affecting real users negatively in their experience with the system. The results of an

offline experiment will be less trustworthy than the ones obtained from an online study because there was no real interaction with the system [128], but they are usually exploited as the first stage in the preparation of the in vivo experimentation.

Even if an offline study is only the first step in the process of evaluating a recommender, it is necessary to rely on a solid evaluation protocol and a set of well-defined metrics that are able to capture all the characteristics of the analyzed algorithm [47]. In the following, we will introduce an offline evaluation framework for sequence-based recommender systems that we called Sequeval.

Our framework is based on the concept of sequence as formalized in Section 4.1. First, an initial dataset is transformed into a set of sequences following Algorithm 4.1. Then, the sequences are split between training and test sets. At this point, one or more external recommenders are plugged into the framework: they are exposed to the training sequences and they are asked to create suggested sequences starting from the same seeds of the test ones. A possible strategy for suggesting additional sequences is described in Algorithm 4.2, but the user can define other approaches. Finally, considering the recommendations available, the framework can compute different evaluation metrics.

In details, Sequeval is made of an evaluation protocol, presented in Section 4.2.1, a set of evaluation metrics, described in Section 4.2.2, and a software implementation that is introduced in Section 4.2.3.

4.2.1 Evaluation Protocol

One of the first problems that an evaluation framework should consider is how to split the dataset between the training set and the test set. This task is not trivial, as it will deeply influence the outcome of the experimentation [116]. Since we are dealing with sequences, we need to split the set of sequences $\mathcal{S}$ in a training and test set such that $\mathcal{S} = \mathcal{S}_{training} \cup \mathcal{S}_{test}$.

Several solutions to this problem are possible: a simple but effective one is to perform the splitting by randomly assigning sequences to these sets according to a certain ratio, typically the 80% for training and the 20% for testing. If the number of sequences available is limited, it is necessary to perform a cross-validation. Another possibility is to identify an appropriate point in time and to consider all the sequences created before it as part of the training set, and after it as part of the test set. This protocol simulates the behavior of a recommender introduced at that point in time and it avoids too optimistic results caused by the knowledge of future events [101]. The latter solution can be considered the most reliable one, but if we do not have any temporal information because the sequences have already been created, it is necessary to adopt a random protocol.

More in general, it is impossible to identify a splitting method that is appropriate for every experiment, as it depends on the domain and on the dataset available. For this reason, Sequeval does not impose the adoption of a particular splitting

protocol, but the experimenter can choose the most appropriate one.

To compute the metrics that are part of the evaluation framework, the sequence-based recommender is trained with all the sequences $\mathbf{s} \in \mathcal{S}_{training}$. Then, for each test sequence $\mathbf{s} \in \mathcal{S}_{test} : \mathbf{s} = \langle \mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_n \rangle$, we predict a recommended sequence $\overline{\mathbf{s}}$ of length k using $\mathbf{r}_1 = (\iota_1, \upsilon, \tau_1)$ as seed rating, i.e., $\overline{\mathbf{s}} = \text{sequence}(\mathbf{r}_1, k)$. Therefore, $\overline{\mathbf{s}}$ is a sequence suggested by the recommender, for the same user and starting from the same item of $\mathbf{s}$. We also define $\mathbf{s}'$ as the reference sequence, i.e., $\mathbf{s}' = \langle \mathbf{r}_2, \mathbf{r}_3, \ldots, \mathbf{r}_n \rangle$ or $\mathbf{s}' = \mathbf{s} - \langle \mathbf{r}_1 \rangle$. The reference sequence is equal to the original sequence, but the first rating is omitted, as it was already exploited for creating the recommended sequence. This procedure is graphically illustrated in Figure 4.1.

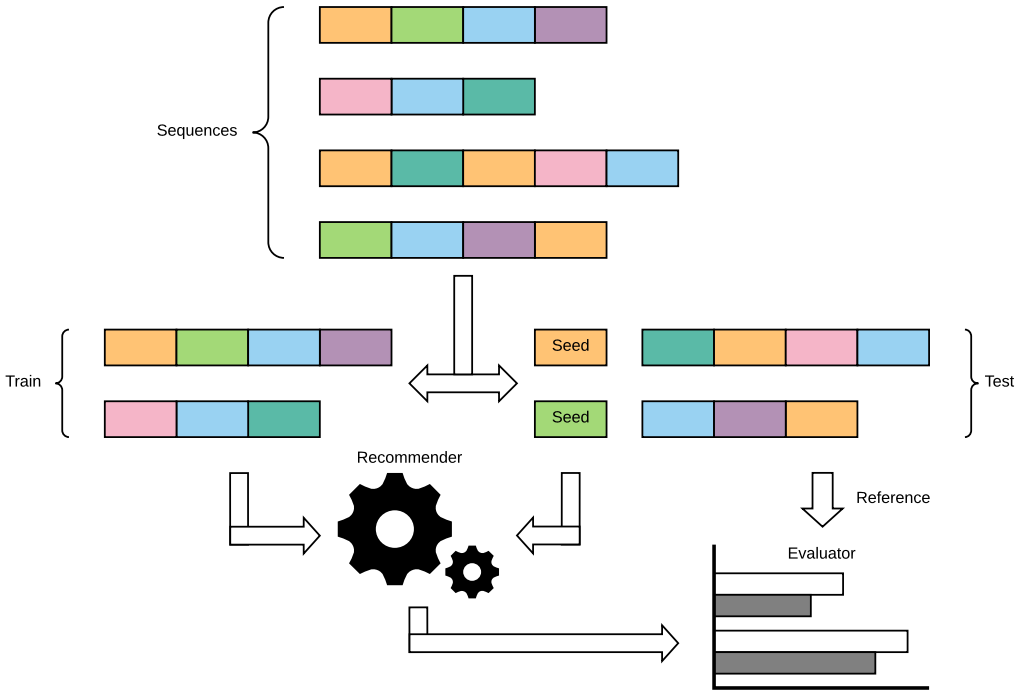


Figure 4.1: An illustration of the evaluation procedure. First, the set of sequences is split between training and test set. Then, the recommender is trained with the sequences available in the training set. Finally, the recommender is asked to generate a sequence for each seed from the test set; such sequences are compared with the corresponding reference sequences.

4.2.2 Evaluation Metrics

The second component of Sequeval is a set of eight metrics that we present in the following. In order to address RQ2.2, we include in such set not only classic metrics such as coverage and precision but also less widespread ones such as novelty, diversity, and serendipity. Furthermore, we introduce the metric of perplexity, as it

is explicitly designed for characterizing sequences [16]. In contrast, we decided to avoid measuring recall because it is clear that the number of recommended items is often likely to be much lower than the total number of relevant items.

Coverage

In general, the coverage of a recommender is a measure that captures the number of items in the catalog over which the system can make suggestions [55]. For example, in an online store scenario, it could represent the percentage of products that are recommended to users in a certain period of time. An algorithm with a higher coverage is generally considered more useful because it better helps users to explore the catalog.

We generate a set of recommended sequences considering as seed the first rating of all sequences in the test set $\mathcal{S}_{test}$ for a recommender that suggests sequences of length k. Afterward, we compute the distinct number of items available in the sequences created and we divide the result by the cardinality of the set $\mathcal{I}$.

$$\text{coverage}(k) = \frac{|\bigcup_{\mathbf{s} \in \mathcal{S}_{test}} \mathcal{I}_{\text{sequence}(\mathbf{r}_1, k)}|}{|\mathcal{I}|} \tag{4.5}$$

This metric expresses the percentage of items that the sequence-based recommender can suggest when generating sequences similar to the ones available in the test set and it is strictly related to its cardinality. This approach is similar to the metric of prediction coverage described by Herlocker et al. [62].

Precision

Precision is a widespread metric in the context of information retrieval evaluation [113] and it represents the fraction of retrieved documents that are relevant. For a traditional recommender system, precision measures the fraction of recommended items that are relevant for a certain user [117]. If we consider a sequence-based recommender, it is necessary to compute this metric for each sequence $\mathbf{s} \in \mathcal{S}_{test}$, instead of each user.

$$\text{precision}(k) = \frac{1}{|\mathcal{S}_{test}|} \cdot \sum_{\mathbf{s} \in \mathcal{S}_{test}} \frac{\text{hit}(\mathbf{s}', \overline{\mathbf{s}})}{\min(|\mathcal{R}_{\mathbf{s}'}|, k)} \tag{4.6}$$

The function $hit : \mathcal{S} \times \mathcal{S} \to \mathbb{N}$ returns the number of items in $\overline{\mathbf{s}}$ that are also available in $\mathbf{s}'$. If the same item is present in $\overline{\mathbf{s}}$ multiple times, it is considered a hit only if it is repeated also in $\mathbf{s}'$. This is an extension to the traditional definition of precision that also considers the fact that an item may appear multiple times inside a sequence.

The number of relevant items is divided by the minimum number between the length of the reference sequence $|\mathcal{R}_{\mathbf{s}'}|$ and the length of the recommended sequence

k. We decided to adopt this solution to avoid penalizing an algorithm that is evaluated considering reference sequences shorter than the recommended sequences.

nDPM

The Normalized Distance-based Performance Metric (nDPM) was originally proposed by Yao in the context of information retrieval [135]. The intuition of the author is that in order to compare a system ranking with a reference user ranking, it is necessary to consider all the possible pairs of items available in the system ranking: they can be agreeing, contradictory, or compatible with respect to the user ranking. We decided to adopt such a metric instead of the Normalized Discounted Cumulative Gain (nDCG) [71] because, in a sequence of recommendations, it is not necessarily true that the first items are more important than the last ones.

$$\text{nDPM}(k) = \frac{1}{|\mathcal{S}_{test}|} \cdot \sum_{\mathbf{s} \in \mathcal{S}_{test}} \frac{2\,\text{pairs}^{-}(\mathbf{s}', \bar{\mathbf{s}}) + \text{pairs}^{u}(\mathbf{s}', \bar{\mathbf{s}})}{2\,\text{pairs}(\bar{\mathbf{s}})} \tag{4.7}$$

The function $pairs^{-} : \mathcal{S} \times \mathcal{S} \to \mathbb{N}$ returns the number of pairs in the sequence $\bar{\mathbf{s}}$ that are in the opposite order with respect to the reference sequence $\mathbf{s}'$. The function $pairs^{u} : \mathcal{S} \times \mathcal{S} \to \mathbb{N}$ returns the number of pairs in the sequence $\bar{\mathbf{s}}$ for which the ordering is irrelevant, i.e., when at least one of the items is not available in $\mathbf{s}'$ or when at least one of the items is available multiple times in $\mathbf{s}'$. Finally, the function $pairs : \mathcal{S} \to \mathbb{N}$ returns the number of all possible pairs available in the recommended sequence $\bar{\mathbf{s}}$. The pairs are created without considering the ordering of the items inside a pair: for example, if we have the sequence $\langle a, b, c \rangle$, the possible pairs are $(a, b), (a, c), (b, c)$.

The value of this metric will result close to 1 when the sequences generated by the recommender are contradictory, to 0 when they have the same ranking, and to 0.5 when the ordering is irrelevant because they contain different items. A low precision will imply a nDPM very close to 0.5.

Diversity

The metric of sequence diversity included in this framework is inspired by the metric of Intra-List Similarity proposed by Ziegler et al. [141]. The recommended sequences are considered to be lists of items and the obtained value is not related to their internal ordering. The purpose of this metric is understanding if the sequences contain items that are sufficiently diverse. A higher diversity may be beneficial for the users, as they are encouraged to better explore the catalog [97].

$$\text{diversity}(k) = \frac{1}{|\mathcal{S}_{test}|} \cdot \sum_{\mathbf{s} \in \mathcal{S}_{test}} \frac{\sum_{\forall i, \forall j : 0 < i < j}^{k} 1 - \text{sim}(\bar{\imath}_i, \bar{\imath}_j)}{k \times (k-1)} \tag{4.8}$$

The function $sim : \mathcal{I} \times \mathcal{I} \to [-1, 1]$ is a generic similarity measure between two items. This measure may be taxonomy-driven or content-based: for example, a possible content-based similarity measure is the cosine similarity. The resulting value is a number in the interval $[0, 2]$: higher values represent a higher diversity.

Novelty

Vargas et al. [132] suggested that it would be useful to be able to characterize the novelty of the recommendations. They proposed a metric that rewards algorithms capable of identifying items that have a low probability of being already known by a specific user because they belong to the long-tail of the catalog. We have included such metric in our framework to assess whether the items available in the suggested sequences are not too obvious.

$$\text{novelty}(k) = -\frac{1}{|\mathcal{S}_{test}| \times k} \cdot \sum_{\mathbf{s} \in \mathcal{S}_{test}} \sum_{i=1}^{k} \log_2 \text{freq}(\bar{\iota}_i) \tag{4.9}$$

The function $freq : \mathcal{I} \to [0, 1]$ returns the normalized frequency of a certain item $\iota \in \mathcal{I}$, i.e., the probability of observing that item in a given sequence $\mathbf{s} \in \mathcal{S}_{training}$. We can define the probability of observing the item ι as the number of ratings related to ι in the training sequences divided by the total number of ratings available. We also assume that $\log_2(0) \doteq 0$ by definition, to avoid considering as novel items for which we do not have any information, i.e., the items that do not appear in the training sequences.

Serendipity

Serendipity can be defined as the capability of identifying items that are both attractive and unexpected [48]. Ge et al. proposed to measure the serendipity of a recommender by relying on the precision of the generated lists after having discarded the items that are too obvious [47].

To create a list of obvious items, it is possible to exploit a *primitive* recommender that is a recommender only capable of making obvious suggestions. For example, a primitive recommender could be implemented using the Most Popular (MP) baseline, which is defined in Section 4.2.3. It is reasonable to assume that popular items do not contribute to the serendipity of the recommendations because they are already well known by many users.

By modifying the metric of precision described in Section 4.2.2, it is possible to introduce the concept of serendipity in the evaluation of a sequence-based recommender. In this case, the primitive recommender will always create a sequence of length k that contains the items that are have been observed with the highest frequency in the training set.

$$\text{serendipity}(k) = \frac{1}{|\mathcal{S}_{test}|} \cdot \sum_{\mathbf{s} \in \mathcal{S}_{test}} \frac{\text{hit}(\mathbf{s}', \overline{\mathbf{s}} - \hat{\mathbf{s}})}{\min(|\mathcal{R}_{\mathbf{s}'}|, k)} \tag{4.10}$$

We define $\hat{\mathbf{s}}$ as the sequence generated by the primitive recommender from the same seed of $\overline{\mathbf{s}}$, i.e., $\hat{\mathbf{s}} = \text{primitive}(\mathbf{r}_1, k)$. Moreover, the sequence $\overline{\mathbf{s}} - \hat{\mathbf{s}}$ contains all the ratings related to the items available in $\overline{\mathbf{s}}$ that are not present in $\hat{\mathbf{s}}$. The resulting value will be a number in the interval $[0, 1]$, lower than or equal to precision. The difference between precision and serendipity represents the percentage of obvious items that are correctly suggested.

Confidence

The metric of confidence reflects how much the system trusts its own suggestions and it is useful for understanding how robust the learned model is [61]. It is usually computed as the average probability that the suggested items are correct. This metric expresses the point of view of the recommender, as the probability is reported by the model. Therefore, the metric is always equal to 1 with the MP recommender, as it is certain of the predictions.

A sequence-based recommender generates the next item of the sequence by considering all the previous items. For this reason, we can interpret the conditional probability of obtaining a certain item, given the sequence of previous ones, as the confidence that the system has in that suggestion.

$$\text{confidence}(k) = \frac{1}{|\mathcal{S}_{test}| \times k} \cdot \sum_{\mathbf{s} \in \mathcal{S}_{test}} \sum_{i=1}^{k} P(\overline{\iota}_i | \overline{\iota}_{i-1}, \overline{\iota}_{i-2}, \dots) \tag{4.11}$$

We also define $\overline{\iota}_0 \doteq \iota_1$, i.e., the *zero*-th item of the recommended sequence is its seed item. Therefore, this metric is computed by also considering the probability of obtaining the first item $\overline{\iota}_1$, given the seed item of $\overline{\mathbf{s}}$.

Perplexity

Perplexity is a widespread metric in the context of neural language modeling evaluation [16], typically used to measure the quality of the generated phrases. Because there is a strong similarity between creating a sequence of natural language words and sequence of recommended items given an initial seed, perplexity can be also successfully exploited in this context.

This metric can be defined as the exponential in base 2 of the average negative log-likelihood of the model, i.e., the cross-entropy of the model. For models based on the cross-entropy loss function such as neural networks, the perplexity can also be seen as a measure of convergence of the learning algorithm. Differently from the metric of confidence, the conditional probability $P(\iota_{i+1} | \iota_i, \iota_{i-1}, \dots)$ is computed

considering the items of the test sequence $\mathbf{s}$, and not of the recommended sequence $\bar{\mathbf{s}}$. For this reason, it does not express the point of view of the recommender.

$$\text{perplexity} = 2^{-\frac{1}{\sum_{\mathbf{s}\in\mathcal{S}_{test}}|\mathcal{R}_{\mathbf{s}}|-1}\cdot\sum_{\mathbf{s}\in\mathcal{S}_{test}}\sum_{i=0}^{|\mathcal{R}_{\mathbf{s}}|-1}\log_2 P(\iota_{i+1}|\iota_i,\iota_{i-1},\dots)} \tag{4.12}$$

Intuitively, the obtained value represents the number of items from which an equivalent random recommender should choose to obtain a similar sequence. The lower is the perplexity, the better is the system under evaluation. Therefore, the perplexity of a random recommender is equal to $|\mathcal{I}|$. If the performance of the recommender is worse than a random one, the perplexity will be higher than $|\mathcal{I}|$: for example, if only one conditional probability is equal to zero, then perplexity $= +\infty$.

4.2.3 Implementation

The third component of Sequeval is `sequeval` [93], a Python implementation of the evaluation framework that is publicly available on GitHub.[1] This implementation is based on the protocol presented in Section 4.2.1 and it includes the metrics described in Section 4.2.2.

In details, `sequeval` is a Python package designed following a modular structure, which is graphically represented in Figure 4.2. For each component, we defined an abstract class and then we realized one or more possible implementations to enable software extensibility.

The *loader* module is in charge of reading an input file containing user ratings. We have implemented a concrete *loader* capable of processing a textual file in a MovieLens-like format (UIRT), but the support to other formats can be easily added. It is optionally possible to ignore users or items that do not have a minimum number of ratings, in order to avoid data sparsity issues. The *builder* module creates the sequences of items from the initial ratings: ratings from the same user that are distant in time less then a threshold are grouped inside the same sequence. Ratings that do not belong to any sequence are discarded.

The *profiler* module computes some statistics about the generated sequences, for example their average length. The *splitter* module assigns the sequences created by the *builder* to the training and the test sets, according to a random or to a more realistic timestamp-based strategy. It is up to the experimenter deciding the percentage of sequences in the test set.

The *recommender* module includes an abstract class that needs to be implemented by any recommender that relies on this framework, based on the sequence generation logic formalized in Algorithm 4.2. The purpose of the abstract class in the *similarity* module is to compute a content-based similarity metric between two

[1] https://github.com/D2KLab/sequeval

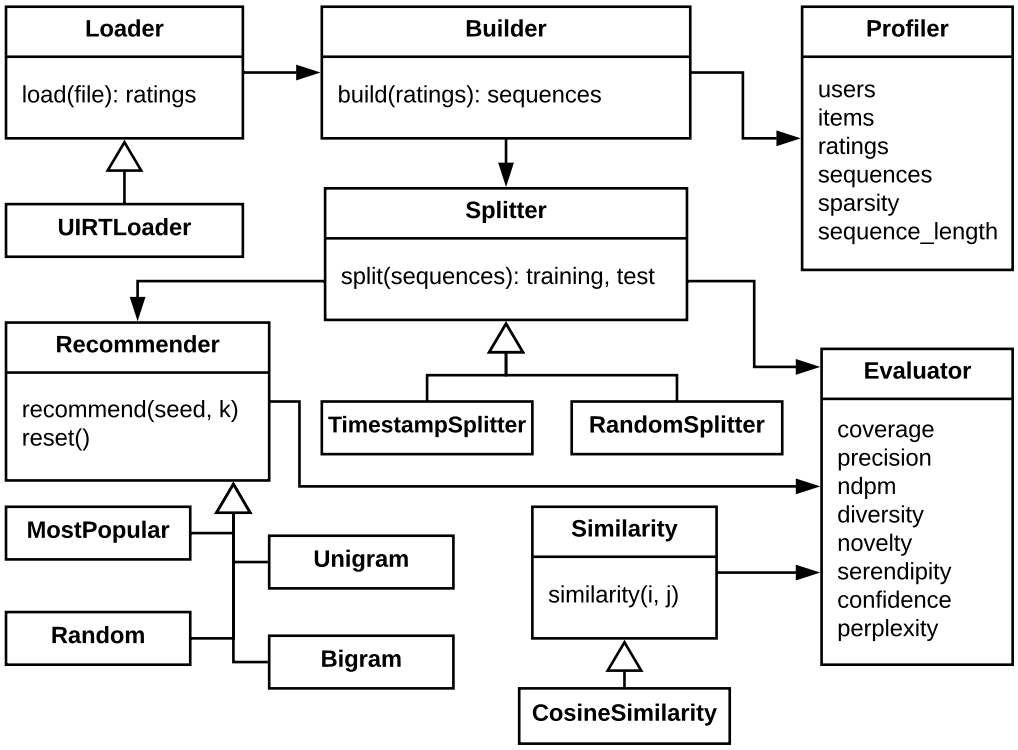


Figure 4.2: A simplified UML class diagram of `sequeval`.

items; we have chosen to implement it as a generic cosine-based similarity. Finally, the *evaluator* module computes the measures detailed in Section 4.2.2.

To exploit the proposed framework, it is necessary to realize an implementation of the abstract recommender that must be capable, given the user and the current item of the sequence, of predicting the probabilities for the possible items of being the next one inside the recommended sequence. If the weighted random sampling generation logic is not appropriate, it is possible to override the relative method and to define an alternative recommendation approach.

For obtaining the experimental results, it is necessary to write an evaluation script that relies on this library. We provide a simple evaluation script which can be used to perform different experiments. This script can be easily modified to accommodate novel recommendation techniques.

We also created an extensive test suite achieving the 98% of code coverage for validating the robustness of our implementation and for better supporting future improvements and developments.

For demonstrative purposes, we have implemented four baseline recommenders, which are illustrated in the following and represent our answer to RQ2.3. These baselines can be interpreted as an adaptation of classic non-personalized recommendation techniques to our sequence-based scenario.

Most Popular The MP recommender analyzes the sequences available in the training set to compute the popularity of each item, i.e., the number of times an item appears in the training sequences. Then, at recommendation time, it ignores the seed rating, and it always creates a sequence that contains the MP item as the first rating, the second MP item as the second rating, and so on. More formally, the probability that the item ι_i will appear in the i-th rating of the sequence is $P(\iota_i) = 1$, where i also represents the position of the item in the ranking of the MP ones.

Random The random recommender simply creates sequences composed of ratings that contain an item randomly sampled from a uniform probability distribution. The seed rating is discarded and the probability of observing the item ι_i is $P(\iota_i) = 1/|\mathcal{I}|$, where $|\mathcal{I}|$ represents the number of items available.

Unigram The unigram recommender can generate sequences that contain ratings with items sampled with a probability proportional to the number of times they were observed in the training sequences. In particular, the probability of observing the item ι_i is equal to the number of ratings containing ι_i divided by the total number of ratings available in the training sequences. As with the previous baselines, the seed rating is ignored during the recommendation.

Bigram The bigram recommender estimates the 1-st order transition probabilities among all possible pair of items available in the training sequences. The add-one smoothing technique is exploited to avoid the attribution of a strict zero probability to the pairs that were not observed during the training phase [29]. At recommendation time, the seed rating is exploited for selecting the first item, and then each item will influence the choice of the next one. The probability of sampling item ι_i after item ι_{i-1} is equal to the number of times this transition occurred in the training sequences plus one divided by the total number of transitions available.

4.3 Experimental Analysis

In this section, we perform an experimental analysis of Sequeval by relying on its implementation described in Section 4.2.3 for comparing the behavior of the four baselines with a recommender system based on Conditional Random Fields (CRF) [125] and another one that exploits Recurrent Neural Networks (RNN) [52]. The purpose of this comparison is to assess the validity of the framework by conducting an offline evaluation in a realistic scenario. Furthermore, we aim to investigate the efficiency of the proposed approach by analyzing the amount of time required to compute the numerical scores per each recommender system, considering datasets of different sizes.

4.3.1 Experimental Setup

The main parameters that need to be specified according to our evaluation framework are the $\delta\tau$ value used to generate the sequences, the splitting protocol, and the length of the recommended sequences k. The $\delta\tau$ value and the splitting protocol depend on the dataset and they are reported in Section 4.3.2. For performing this empirical analysis, we have decided to exploit the 80% of the dataset for training the recommenders and the remaining 20% for testing purposes. The length of the recommended sequences depends on the specifications of the target application: for this evaluation, we have chosen to set $k = 5$.

To compute the metric of diversity, we have selected the cosine similarity among the training sequences as the proximity measure between two items. In fact, we assume that two items are similar if they appear the same number of times inside the same training sequences. Furthermore, we have assumed that if an item is unknown, its similarity with another one is *zero* by definition.

In the following, we provide the rationale for the usage and the implementation details of the two recommenders based on CRF and RNN.

CRF We have implemented a CRF-based recommender system using the `CRFsuite` software package.[2] Since we are interested in predicting an item given the previous one, we have considered to be feature vectors the training sequences without their last rating and as corresponding output vectors the same sequences without their first rating. We have used the gradient descent algorithm with the L-BFGS method [96] as the training technique. We have chosen to generate both the state and the transition features that do not occur in the dataset and we have set the maximum number of iterations allowed for the optimization algorithm to 100.

RNN We have also experimented with a sequence recommender, originally designed for the tourism domain, based on RNN [100] that are specifically meant to deal with sequential data. The hyper-parameters of the network have been optimized through a manual search on the validation set in [100], obtaining: n_layers = 3, dropout = 0.2, learning_rate = 0.0001, n_hidden = 64, and n_epochs = 10. The main difference of RNNs with respect to standard feed-forward neural networks is the presence of a hidden state variable h_t, whose value depends both on the input data presented at time x_t and on the previously hidden state h_{t-1} [52] using loop connections. A typical application of RNNs in neural language modeling is the generation of text by recursively applying a "next word prediction" [124]. In the same spirit, we address the problem of next item prediction. The probability of the next rating given the

[2] http://www.chokkan.org/software/crfsuite

previous ones $P(\mathbf{r}_k|\langle \mathbf{r}_0, \mathbf{r}_1, \ldots, \mathbf{r}_{k-1}\rangle)$ is learned during the training process of the neural network without the need for specifying a particular memory window as in Markov models.

For conducting this experimental campaign, we relied on a machine equipped with two 12-cores Intel Xeon processors (E5-2680 v3 at 2.50 GHz) and 128 GB of RAM. However, note that `sequeval` is a single-threaded application and its memory requirements are actually much lower, around 2.5 GB with the most demanding dataset at our disposal.

4.3.2 Datasets

We have performed the experimental analysis considering two different datasets, namely Yes.com and Foursquare. The former is related to the music domain, while the latter deals with check-ins performed at specific POIs.

Because we are interested in modeling sequences, it is important that the temporal information available is actually meaningful. For example, the popular MovieLens datasets [57] cannot be exploited for our purposes because the timestamps are associated with the action of assigning a rating on the platform and not with the action of watching a movie. This hypothesis is supported by the fact that, if we apply Algorithm 4.1 to the MovieLens 1M dataset with $\delta\tau = 1$ h, we obtain unrealistic sequences with an average length of about 56 movies.

The Yes.com and Foursquare datasets are characterized by a different distribution of their items, i.e., songs and venue categories, as it can be observed from Figure 4.3. In particular, Foursquare contains few items that are extremely popular, while Yes.com presents a plot that is smoother. This conclusion is numerically supported by the values of entropy [118] obtained for the two distributions, which are 4.95 for Foursquare and 6.75 for Yes.com. Furthermore, the number of sequences available in Foursquare is about 40 times higher with respect to Yes.com. Table 4.1 summarizes the number of users, items, ratings, and sequences available in these datasets, which are described in detail in the following sections.

Dataset	$\lvert\mathcal{U}\rvert$	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{R}\rvert$	$\lvert\mathcal{S}\rvert$
Yes.com	1	1089	118,022	10,551
Foursquare	44,319	651	1,047,429	400,261

Table 4.1: The number of users, items, ratings, and sequences after the preprocessing steps. The Yes.com dataset has only one user.

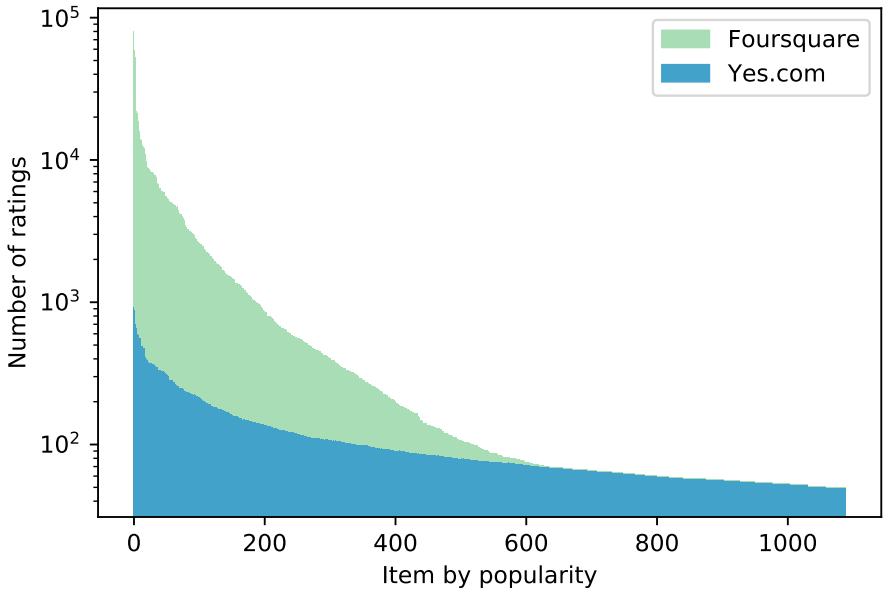


Figure 4.3: A stacked bar plot with a logarithmic scale representing the number of ratings for each item. Note the different shapes of their long-tail distributions: it is possible to observe that Foursquare has more popular items than Yes.com.

Yes.com

This dataset contains several playlists originally collected by Shuo Chen from Yes.com in the context of his research on Metric Embedding [28]. Such website provided a set of APIs[3] for programmatically retrieving songs aired by different radio stations in the United States. By crawling them in the period from December 2010 to May 2011, he managed to obtain 2,840,553 transitions. Even if Yes.com is no longer active, the playlist dataset is publicly available.[4]

Yes.com does not include the timestamps, but only the playlists. Therefore, we have assumed that each playlist represents a sequence, as defined in our evaluation framework. In this case, it is not necessary to apply Algorithm 4.1 because the sequences are already available in the dataset in an explicit form. Because a timestamp-based splitting is not feasible, we have selected, for this dataset, a random splitting protocol for dividing the sequences between training and test set.

Since we do not have any information regarding the radio stations, it is necessary to consider the playlists as if they were created by the same user. This

[3] http://web.archive.org/web/20150316134941/http://api.yes.com

[4] https://www.cs.cornell.edu/~shuochen/lme/data_page.html

approximation is acceptable in the context of sequence recommendation and it is allowed by the evaluation framework. In fact, differently from traditional evaluation approaches, all the metrics that we propose are averaged over the sequences and not over the users.

Because of the computational complexity of the task, we have randomly reduced the complete dataset 10 times its original size and we have pruned the songs that appear less than 50 times.

Foursquare

The second dataset that we have selected for performing the experimental evaluation of the framework is similar to the one described in [100] and it was created following the same protocol.

We collected the check-ins performed by the users of the Foursquare Swarm mobile application[5] and publicly shared on Twitter from the Twitter API. Then, we retrieved the category of the place associated with the check-in thanks to the Foursquare API. For this reason, the items of the dataset are represented by the venue categories available in the Foursquare taxonomy.[6] The collection phase lasted from October to December 2017.

To avoid exploiting the interactions generated by automated scripts, we have discarded the users that performed multiple check-ins in less than one minute. We have also pruned the check-ins associated with the venue categories that are usually not of interest for a tourist, for example the ones related to workplaces. For generating the sequences more efficiently, we decided to also remove the users that have performed less than 10 check-ins in total.

We have set the $\delta\tau$ parameter of the evaluation framework to 8 h. Regarding the splitting protocol, we have selected the timestamp-based one, considering the timestamp associated with the first rating as the timestamp of the sequence.

4.3.3 Results

Table 4.2 summarizes the figures of the evaluation conducted with Yes.com. The MP recommender achieved a fair precision, but at the price of a very low coverage, because its predictions are deterministic. Unsurprisingly, the lowest precision and the highest novelty and diversity are associated with the random recommender. In contrast, the unigram, the bigram, and the CRF recommenders obtained comparable scores of precision, but the bigram is the most appealing of these three techniques, because of its lower perplexity and higher novelty.

[5] https://www.swarmapp.com

[6] https://developer.foursquare.com/docs/resources/categories

Metric	MP	Random	Unigram	Bigram	CRF	RNN
Coverage	0.0046	1.0000	0.9945	1.0000	0.9991	0.9458
Precision	0.0503	0.0090	0.0127	0.0103	0.0190	0.0782
nDPM	0.5007	0.5000	0.5000	0.5000	0.5000	0.4986
Diversity	0.6925	0.9900	0.9815	0.9854	0.9788	0.9052
Novelty	7.2383	10.380	9.7349	10.315	9.8449	9.5762
Serendipity	0.0000	0.0089	0.0107	0.0095	0.0179	0.0706
Confidence	1.0000	0.0009	0.0016	0.0011	0.0020	0.0123
Perplexity	$+\infty$	1089.0	848.96	637.53	747.33	183.49

Table 4.2: The results of the baselines and both CRF and RNN with Yes.com.

We can observe that the RNN recommender achieved the highest precision and the lowest perplexity, resulting to be the most promising algorithm for future online experimentations. Its nDPM is slightly lower than 0.5, meaning that the items are usually predicted in the correct order. We can also observe that its serendipity is close to the value of precision: for this reason, it is possible to assume that most of the sequences are not obvious.

Table 4.3 lists, instead, the results obtained with Foursquare. In this case, the MP recommender system accounted for the highest precision, meaning that the top-5 items are extremely widespread, but, as usual, its coverage is very limited, and it achieved the lowest novelty. On the other hand, the random recommender scored the lowest precision, and the highest coverage and novelty. The differences among the unigram, the bigram, and the CRF recommenders are more striking than in the previous experiment: with this dataset, the unigram accounted for higher precision because of the popularity of some items, while the bigram for the lowest perplexity.

Metric	MP	Random	Unigram	Bigram	CRF	RNN
Coverage	0.0077	1.0000	0.9616	1.0000	0.9677	0.5069
Precision	0.2259	0.0080	0.0774	0.0607	0.0754	0.0962
nDPM	0.4998	0.5000	0.4994	0.4998	0.4993	0.4991
Diversity	0.9194	0.9971	0.9616	0.9777	0.9621	0.9469
Novelty	4.6056	12.300	7.1421	9.0216	7.3710	6.8374
Serendipity	0.0000	0.0060	0.0256	0.0230	0.0252	0.0365
Confidence	1.0000	0.0015	0.0171	0.0140	0.0179	0.0264
Perplexity	$+\infty$	651.00	141.41	122.99	147.49	140.39

Table 4.3: The results of the baselines and both CRF and RNN with Foursquare.

The RNN recommender system obtained the second-best precision and perplexity, resulting in a good compromise if we are interested in optimizing both these metrics. Its fair coverage and the low value of serendipity are other hints of the fact that the Foursquare dataset contains few items that are very popular: this characteristic was, in fact, learned and exploited by the recommender.

Finally, we report in Table 4.4 the amount of time needed for computing the previously described evaluation metrics per recommendation algorithm with the Foursquare and Yes.com datasets. We observe that in the worst case, the evaluation framework was able to conduct an experimental campaign in a few hours. The metric of diversity was the most computationally expensive one because of the time needed to compute the cosine similarity. This fact is particularly evident if we consider the seconds spent to evaluate the random recommender with the Foursquare dataset.

Dataset	Div.	MP	Random	Unigram	Bigram	CRF	RNN
Yes.com	Yes	0.84	4.25	3.97	4.05	963.65	66.04
Yes.com	No	0.78	1.14	0.98	1.05	865.17	60.68
Foursquare	Yes	15.88	1326.31	58.64	101.73	4096.78	1223.26
Foursquare	No	13.75	24.63	13.05	16.78	3989.17	1194.52

Table 4.4: The time in seconds required to evaluate different algorithms with our framework. We do not consider the time for training the CRF and RNN models. To improve the efficiency of the framework it is possible to avoid computing the computationally expensive metric of diversity.

As expected, the baseline recommenders are less demanding with respect to the CRF and RNN models. However, this analysis is beyond the scope of this work, as we are only interested in optimizing the evaluation framework. If we do not consider the metric of diversity, we observe that the time required for the evaluation phase is linear with respect to the size of the dataset.

4.3.4 Discussion

In the following, we will analyze the results of the empirical analysis to justify the answers to the research questions that were provided in Section 4.1 and in Section 4.2. In particular, our main aim is to explain why it is necessary to rely on a multicriteria framework that includes several metrics to evaluate a sequence-based recommender system.

In Section 4.1 we have introduced the concept of rating and we have defined it considering three different elements: an item, a user, and a timestamp. The idea of associating a user with an item is the basic principle of almost every

recommender, while the timestamp is necessary in order to introduce a temporal dimension, and, therefore, the possibility of creating and suggesting sequences, as proposed in RQ2.1. Nevertheless, we have successfully applied our evaluation framework in an experiment based on the Yes.com dataset, which does not include any user. Even though a more general use case has been considered during its formalization, it is possible to also exploit it in other scenarios, still obtaining an interesting picture of the recommenders under evaluation.

As described in Section 4.2.2, our answer to RQ2.2 is an evaluation framework that includes eight different metrics, capable of capturing the various characteristics of the algorithms available. For example, even if the precision of the MP recommender system is very high when tested with Foursquare, we can immediately discard it because of its low coverage. In the same way, the interesting values of diversity and novelty achieved by the random recommender are associated with an unacceptable score of perplexity.

In Table 4.5 we present an interpretation of the metrics available in the framework. These descriptions are meant to offer a human understanding of the results of the offline evaluation. It is worth noticing that these metrics consider only some of the properties of a recommender system [55]. However, it is our opinion that those properties are the most salient ones that can be analyzed in our context, without realizing a live system.

Metric	Interpretation
Coverage	The percentage of items that are recommended in the evaluation
Precision	The percentage of items that are correctly recommended
nDPM	The correctness of the ordering inside the sequences
Diversity	How diverse are the items inside the sequences
Novelty	How unexpected are the recommended items
Serendipity	The percentage of non-obvious items that are correct
Confidence	The confidence that the recommender has about its predictions
Perplexity	How much the recommender is “surprised” by the test sequences

Table 4.5: A human readable interpretation of the metrics.

The different characteristics of the datasets exploited during the empirical analysis are reflected in their respective figures. In particular, while the values of precision obtained by the random recommender in the two experiments are comparable, the figures associated with both MP and unigram methods are dramatically different. This fact suggests that Foursquare contains a few items that are extremely popular, as it was already clear from Figure 4.3.

On the other hand, the RNN recommender obtained, with both datasets, comparable values of precision, but a very different coverage. For this reason, we can suppose that this approach, differently from the CRF recommender, is capable of

better adapting itself to the characteristics of the dataset. The fact that we can draw such a conclusion supports the validity of Sequeval.

Furthermore, we have observed that the amount of time required to compute the evaluation metrics is linear with respect to the dataset size, if we do not consider the metric of diversity. In fact, the computational cost of the cosine similarity was too elevated when we analyzed the behavior of the random recommender with a more demanding dataset. However, because of the modular structure of `sequeval`, it is easy to avoid computing the metric of diversity for such a recommender.

In line with RQ2.3, we have described in Section 4.2.3 four different baseline recommenders. From the results of the experiments, it is possible to observe that the values obtained by some of them are fixed. For example, the MP recommender will always achieve a serendipity equal to 0, and a confidence equal to 1. Its perplexity is usually $+\infty$, if at least one of the recommended sequences is incorrect. The items suggested are, in fact, considered obvious by definition, and the algorithm is certain of the recommendation because its behavior is deterministic. In a similar way, the perplexity of the random recommender is equal to the total number of items available, i.e., $|\mathcal{I}|$, because of the definition of perplexity provided in Section 4.2.2.

These two baselines are methods commonly exploited in the literature for evaluating recommender systems. Additionally, we have proposed two techniques better suited for the sequence recommendation problem. The unigram recommender is similar to the MP one, but it is non-deterministic, and it obtained a higher novelty. In contrast, the bigram recommender is the most complex baseline, because it considers the previous item to suggest the next one. For this reason, it always achieved the lowest perplexity among the baselines considered.

4.4 Conclusion

In this chapter, we have discussed the problem of recommending sequences of items tailored to the needs of a certain user. We have introduced an offline evaluation framework, called Sequeval, capable of handling this novel family of recommender systems in an offline scenario and we have developed an implementation of it that is publicly available on GitHub. We have included in such a framework an evaluation protocol and eight different metrics, to better capture the characteristics of the algorithms considered.

We have performed an empirical analysis of Sequeval by relying on it for conducting a comparison among four baselines, a CRF recommender, and an RNN-based one. The results have highlighted the fact that this framework is flexible, as it can be successfully applied in non-standard recommendation scenarios, such as with Yes.com, and complete, because of the different metrics included that consider several dimensions of the recommended sequences. In addition, we have observed that the RNN recommender system can effectively adapt itself to the characteristics

of the training dataset. This conclusion supports the validity of Sequeval as a tool for conducting an offline experimentation.

The formal definitions provided in Section 4.1 have been conceived as an extension of the seminal works on recommenders capable of recommending sequences. For this reason, it is possible to set the length of the recommended sequences to 1 if we are interested in obtaining a single item. In a similar way, the item included in the seed rating can be exploited in order to set the context of the recommendation, but it can also be ignored if we want a sequence only based on the target user.

Chapter 5

Evaluation of Top-k Recommender Systems

Different authors have empirically demonstrated that offline evaluation protocols in the context of recommender systems have several weaknesses [116]. For example, it is widely known that comparing results obtained in different experimental settings should be done with caution, as the slightest difference in the evaluation protocol may result in measures that are totally inconsistent [70].

Nevertheless, offline experiments are extremely important for comparing a large number of candidate algorithms without sustaining the costs of an online evaluation involving too many human subjects [55]. After having pruned the set of available systems, it is however advisable to analyze them in a real environment for obtaining more conclusive results, as discussed in Section 2.2.

In this chapter, we apply the knowledge about multicriteria evaluation we gained from the work presented in Chapter 4 to the more traditional setting of top-k lists of suggested items. We propose a way of overcoming the problem of comparing offline evaluation results obtained from different recommendation algorithms in heterogeneous settings. To this end, we designed and implemented an open source evaluation framework for top-k prediction methods, called RecLab,[1] that is capable of interacting with several recommenders using RESTful APIs.

The responsibilities of the evaluator and the recommender are clearly separated because of the distributed architecture of the system. The evaluator is in charge of building a training set, selecting a set of users to whom recommend the items, and computing the evaluation metrics. On the other hand, the recommender must be capable of predicting a list of the most appropriate items for each user, given the information available in the training set.

The configuration parameters of each experiment are left to the user of the

[1] https://github.com/D2KLab/reclab

toolkit, who is free to choose the dataset, the splitting strategy, the size of the test set, the length k of the recommended lists, and the rating threshold between relevant and irrelevant items. The experiment can be designed and run by simply interacting with a web-based GUI provided by the toolkit. Other researchers can easily plug their recommender systems into the evaluation pipeline by implementing the APIs defined by RecLab and by deploying them on a server. Thanks to this approach, it is possible to compare, in a controlled environment, different algorithms and techniques without necessary disclosing their implementation details. The results of each experiment, along with the respective configuration parameters, are publicly available to support accountability and comparative analyses of the results.

More formally, we aim to provide an answer to the following research questions.

RQ3.1 How can different top-k recommender systems be fairly compared in heterogeneous settings without necessary exposing their algorithms?

RQ3.2 To what extent it is possible to support the reproducibility of the experiments and the accountability of the results?

RQ3.3 How can the availability of different metrics support the experimenter in the interpretation of the obtained results?

The remainder of this chapter is structured as follows. In Section 5.1 we introduce the main design choices behind our evaluation framework, while in Section 5.2 we describe how different recommenders can interact with the evaluator. In Section 5.3 we explain the mathematical details of the metrics computed during the evaluation phase. We present and discuss our results in Section 5.4 and, in Section 5.5, we provide the conclusions.

5.1 RecLab Evaluation Framework

RecLab has been implemented as a distributed web application: its users can setup the experimental environment by simply visiting a web page. This step is crucial for the correct execution of the measurements, as selecting inconsistent or wrong values may lead to results that are extremely difficult to interpret [70].

In details, the experimenter needs to specify the following parameters before starting an evaluation:

- the initial rating dataset;
- the technique used to split the dataset;
- the size of the training and the test set;
- the length k of the lists of recommended items;

- the threshold between negative and positive ratings;
- the list of recommenders to be evaluated.

We included in this evaluation framework three widely used rating datasets: MovieLens 100K,[2] MovieLens 1M,[3] and HetRec LastFM.[4] The MovieLens datasets are among the most popular recommender systems datasets about movies preferences [57], while the last one is particularly interesting as it contains the number of times each user listened to a specific artist on LastFM [22]. Other datasets can be easily added by editing a configuration file.

We provide two different methods for splitting the rating dataset $\mathcal{R}$ in a training set $\mathcal{R}_{train}$ and a test set $\mathcal{R}_{test}$ such that $\mathcal{R} = \mathcal{R}_{train} \cup \mathcal{R}_{test}$. The first one is a random splitting method that assigns each rating $\rho \in \mathcal{R}$ to the test set or the training set according to a probability specified by the user, that is proportional to the expected size of the test set. In general, this method should be the preferred one when no temporal information is available [55]; the default size of the test set is the 20% of the ratings present in the whole dataset.

A second splitting technique is based on the timestamps associated to the ratings: the whole dataset is ordered from the oldest to the newest rating; then, the first ones are assigned to $\mathcal{R}_{train}$, while the others to $\mathcal{R}_{test}$. This protocol simulates the behaviour of a recommender introduced at a certain point in time in the system. While the HetRec LastFM dataset does not include any timestamp, the MovieLens ones do. However, their values probably do not represent when users watched a certain movie, but when they rated it on the platform.

Another fundamental parameter of the experiment is the length k of the lists of recommended items, as it will deeply influence all the metrics computed by the evaluator. This value should be set according to the number of items that the final application will display to its users. Typical values for this criterion are 5 or 10.

The experimenter also needs to specify what is the threshold between negative and positive ratings: only ratings with a value strictly greater than the threshold will be considered *likes* during the evaluation phase. Many recommenders will only analyze positive ratings during the training phase. However, this is beyond the scope of the evaluator, so it is a responsibility of the recommender to properly load the ratings. The most appropriate value for this parameter depends on the dataset: a typical setting for MovieLens datasets is 3, thus only 4 and 5 stars ratings are considered positive. For the HetRec LastFM dataset, as the rating value represents the number of times a user listened to an artist, any small number may be reasonable, including 0.

[2] https://grouplens.org/datasets/movielens/100k/

[3] https://grouplens.org/datasets/movielens/1m/

[4] https://grouplens.org/datasets/hetrec-2011/

For demonstrative purposes, we included in RecLab a set of recommender systems that follow the interaction protocol described in Section 5.2. However, anyone is encouraged to implement other techniques for the purpose of evaluating them with this framework. Further recommenders can be added by simply inserting their URIs in a configuration file present in our repository. All available recommenders are then displayed to the experimenter, for letting her select which ones to evaluate.

In details, we have realized the classical most popular and random recommenders. Our most popular recommender is *personalized*: it will never suggest to a certain user items already rated by the same user in the training set. On the other hand, the random recommender will select any item available in the training set with an equal probability.

Furthermore, we have included the MyMediaLite [46] implementations of the Item KNN, User KNN, BPRMF, and WRMF recommender systems using their default settings.[5] BPRMF is a recommendation algorithm based on a Bayesian ranking optimization method [111], while WRMF is weighted matrix factorization technique [68].

5.2 Interaction Protocol

RecLab is a distributed evaluation framework. For this reason, it exploits consolidated web standards to create a communication channel among itself and the recommenders under analysis: the overall protocol is graphically depicted as a UML sequence diagram in Figure 5.1 and it represents our answer to RQ3.1. This interaction is initiated when the experimenter decides to execute a new evaluation, and it is repeated for every recommender selected as part of it.

First, the evaluator requests the recommender to train a new recommendation model with a POST on the resource `/model`. It provides to the recommender a URI from which it can download the training set and the rating threshold, as specified by the experimenter. Only the ratings with a value strictly greater than the threshold should be considered as positive feedbacks.

The recommender can now retrieve the training set from the provided URI. Each training set is specific for a particular experiment, but it is created at run time by the evaluator using the configuration settings specified by the user. The training set consists of a list of ratings, where each rating associates a user, an item, and, optionally, a timestamp to a numerical value.

Now the recommender has all the information required to perform the training process. Meanwhile, the evaluator will start asking asynchronously to the recommender if this phase has ended with a GET of `/model`. When the recommender is

[5] http://www.mymedialite.net/documentation/item_prediction.html

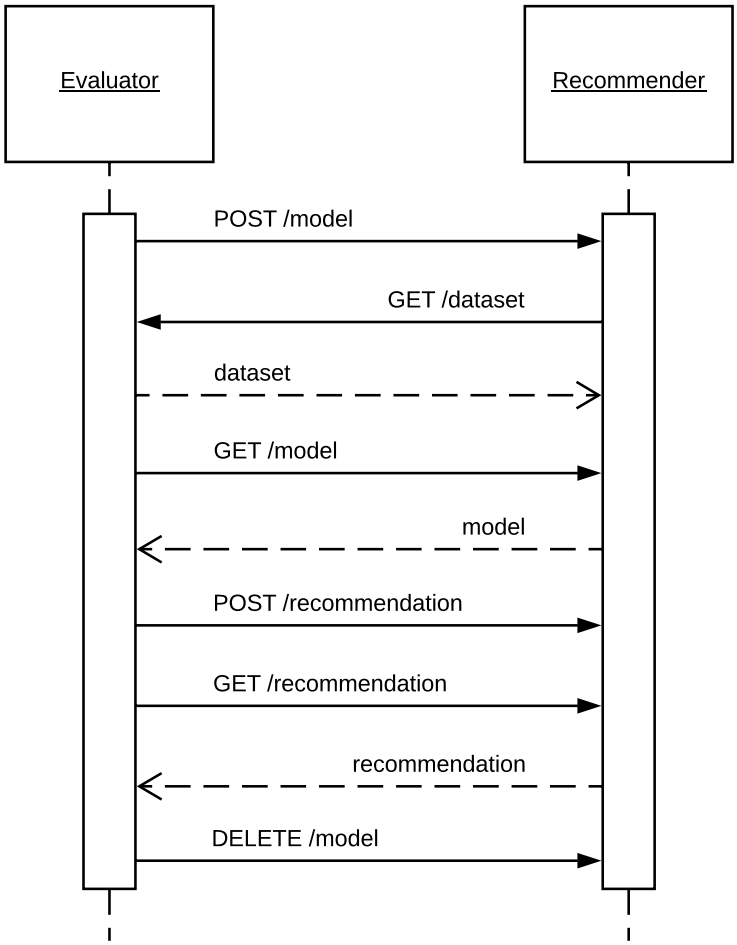


Figure 5.1: A UML sequence diagram describing the RecLab interaction protocol.

ready to suggest items, the evaluator is informed of that and it can proceed to the next step.

The evaluator asks the recommender system, with a POST on the resource `/recommendation`, to create a list of k items for each user specified in the payload of the request. The list of users to whom recommend the items contains all the users available in the test set, while the value of k was initially provided by the experimenter. Note that it is a responsibility of the recommender avoiding to suggest items already rated by a particular user in the training set. In general, there is no guarantee that the test set will only contain users and items available in the training set.

Because also the recommendation phase may be time consuming, it is considered asynchronous, similarly to the training one. The evaluator starts asking with a GET on the same resource if the lists of recommendations are ready. When they are correctly retrieved, the evaluator asks the recommender to DELETE the `/model`

to avoid consuming memory, while it begins to compute the evaluation metrics detailed in Section 5.3.

5.3 Evaluation Metrics

In order to better analyze the recommender systems under evaluation from different perspectives, we decided to include in RecLab a comprehensive set of seven different metrics. In fact, it is not possible to accurately evaluate in an offline experiment a set of recommenders by only relying on a single indicator [62]. In addition to traditional metrics such as coverage and precision, we propose less widespread ones like novelty, diversity, and serendipity.

These metrics are similar to the ones introduced in Section 4.2.2, but while the latter were designed to evaluate sequences of recommended items, the former are more appropriate for ranked lists of suggestions. We added to RecLab the popular metric of recall, which was not really meaningful in the context of sequence-based recommenders. Furthermore, the metric of nDPM was put aside in favor of nDCG, as it seemed more appropriate to characterize the raking of a list. Finally, we did not consider the metrics of confidence and perplexity, as they would require the recommender systems to also provide a probability for each suggested item, which is beyond the scope of our evaluation framework. This multicriteria set is not final, as RecLab can be easily expanded in order to compute additional metrics that the community considers useful.

In the following, we define $\mathcal{U}$ as the set users $\upsilon \in \mathcal{U}$, $\mathcal{I}$ as the set of items $\iota \in \mathcal{I}$, and $\mathcal{R}$ as the set of ratings $\rho \in \mathcal{R}$. Furthermore, we define $rec(\upsilon, k)$ as the list of the top-k items recommended to user υ and $ref(\upsilon)$ as the set of items rated positively by user υ in the test set $\mathcal{R}_{test}$.

5.3.1 Coverage

The coverage of a recommender is a measure that represents the number of items in the catalog over which the system can make suggestions [55]. Given the lists of recommended items for each user in the test set, we compute the percentage of suggested items with respect to the distinct items available in the training set.

$$\text{coverage}(k) = \frac{|\bigcup_{\upsilon \in \mathcal{U}_{test}} \mathcal{I}_{\text{rec}(\upsilon,k)}|}{|\mathcal{I}_{train}|}$$

This metric captures if the recommender is capable of suggesting enough various items to each user, or if it always proposes the same items to all the users. Coverage should be analyzed together with precision, otherwise it is clear that random recommendations will achieve optimal results [62].

5.3.2 Precision

Precision, in the context of information retrieval, represents the fraction of selected documents that are relevant. For a recommender system, it measures the fraction of recommended items that are liked by a user [117].

$$\text{precision}(k) = \frac{1}{|\mathcal{U}_{test}|} \cdot \sum_{v \in \mathcal{U}_{test}} \frac{|rec(v,k) \cap ref(v)|}{k}$$

In order to avoid overestimating the value of precision, we assume that all non-rated items are irrelevant [121].

5.3.3 Recall

Complementary to precision, recall represents the fraction of relevant documents that have been selected. In the context of recommender systems, it measures the fraction of correctly recommended items with respect to all the items the are liked by a user [117].

$$\text{recall}(k) = \frac{1}{|\mathcal{U}_{test}|} \cdot \sum_{v \in \mathcal{U}_{test}} \frac{|rec(v,k) \cap ref(v)|}{|ref(v)|}$$

If the set of items rated positively by a user is empty, we assume that the recall for that user is 0 by definition.

5.3.4 nDCG

The Normalized Discounted Cumulative Gain is another information retrieval metric, that also considers a logarithmic gain related to the position of each correctly predicted item [71]. This metric reveals if a recommender is capable of correctly predicting items at the top of the list.

$$\text{dcg}(k) = \frac{1}{|\mathcal{U}_{test}|} \cdot \sum_{v \in \mathcal{U}_{test}} \sum_{i=1}^{k} \frac{|\{\iota_i\} \cap ref(v)|}{log_2(i+1)}$$

Where $\iota \in rec(v,k)$. The DCG value needs to be divided by the ideal DCG for normalization. The ideal DCG can be computed with the same formula, assuming that all recommended items are relevant for the associated user.

5.3.5 Novelty

This metric rewards algorithms capable of suggesting items that belong to the long-tail of the catalog, and so it is unlikely that they are already known by a certain user [132].

In this way, it is possible to check that the recommended items are not too popular and obvious.

$$\text{novelty}(k) = -\frac{1}{|\mathcal{U}_{test}| \times k} \cdot \sum_{v \in \mathcal{U}_{test}} \sum_{i=1}^{k} \log_2 \text{freq}(\iota_i)$$

Where $\iota \in rec(v, k)$ and $freq : \mathcal{I} \to [0, 1]$ represents the probability of observing the item ι in $\mathcal{I}_{train}$. We also assume that $\log_2(0) \doteq 0$ by definition.

5.3.6 Diversity

The metric of diversity is inspired by the metric of Intra-List Similarity proposed by Ziegler et al. [141]. It measures how much the items included in the recommended lists are diverse. A higher diversity may be beneficial for the users, as they are encouraged to better explore the catalog [97].

$$\text{diversity}(k) = \frac{1}{|\mathcal{U}_{test}|} \cdot \sum_{v \in \mathcal{U}_{test}} \frac{\sum_{\forall i, \forall j: 0 < i < j}^{k} 1 - \text{sim}(\iota_i, \iota_j)}{k \times (k-1)}$$

Where $\iota \in rec(v, k)$ and $sim : \mathcal{I} \times \mathcal{I} \to [-1, 1]$ is a similarity measure between two items. We decided to exploit the cosine similarity computed between the vectors representing the users who liked the two items in the training set.

The resulting value is a number in the interval $[0, 2]$: higher values imply an higher diversity.

5.3.7 Serendipity

Serendipity can be defined as the capability of identifying items that are both attractive and unexpected [48]. It is possible to measure the serendipity of a recommender by computing its precision after having discarded the items suggested by a *primitive* recommender [47].

$$\text{serendipity}(k) = \frac{1}{|\mathcal{U}_{test}|} \cdot \sum_{v \in \mathcal{U}_{test}} \frac{|(rec(v, k) \setminus prim(k)) \cap ref(v)|}{k}$$

Where $prim(k)$ is the set of the top-k most popular items available in the training set. We can, in fact, suppose that popular items are already known by several users, and thus they cannot contribute to the serendipity of the suggestions.

5.4 Experimental Results

To prove the effectiveness of RecLab, we performed three different experiments with the recommender systems described in Section 5.1 whose implementation details are available in our repository, in line with RQ3.2.

Algorithm	Coverage	Precision	Recall	nDCG	Novelty	Diversity	Serendipity
Random	1.000000	0.005152	0.002526	0.005069	13.37526	0.966485	0.005003
Most Popular	0.017920	0.145146	0.084294	0.158512	8.580345	0.600524	0.071869
Item KNN	0.473527	0.212028	0.137608	0.224146	10.55504	0.788987	0.196637
User KNN	0.141732	0.263337	0.189679	0.295034	9.052157	0.657436	0.205550
BPRMF	0.326907	0.225464	0.148515	0.247625	9.473122	0.717023	0.176972
WRMF	0.120554	0.258400	0.169925	0.287808	9.138422	0.667673	0.210835

Table 5.1: Evaluation results with the MovieLens 1M dataset and a random splitting.

Algorithm	Coverage	Precision	Recall	nDCG	Novelty	Diversity	Serendipity
Random	0.993719	0.017555	0.002910	0.017394	13.41860	0.963699	0.016938
Most Popular	0.037411	0.257487	0.053148	0.273653	8.546251	0.528352	0.066517
Item KNN	0.344894	0.231296	0.056491	0.244158	9.759138	0.670575	0.095962
User KNN	0.117422	0.275042	0.062094	0.293720	8.842892	0.567265	0.114470
BPRMF	0.220918	0.265339	0.062845	0.282460	9.083545	0.611382	0.112675
WRMF	0.136537	0.276164	0.065600	0.297178	8.941997	0.587175	0.121929

Table 5.2: Evaluation results with the MovieLens 1M dataset and the timestamp splitting.

Algorithm	Coverage	Precision	Recall	nDCG	Novelty	Diversity	Serendipity
Random	0.708420	0.000584	0.000632	0.000623	15.30801	0.998417	0.000584
Most Popular	0.001692	0.066773	0.069242	0.076932	7.736651	0.632728	0.019161
Item KNN	0.235489	0.126168	0.131249	0.143234	12.56312	0.774870	0.101486
User KNN	0.031299	0.158652	0.164218	0.193206	8.735683	0.717815	0.115711
BPRMF	0.024597	0.078715	0.081908	0.086609	8.280671	0.752627	0.038800
WRMF	0.015617	0.164809	0.170302	0.201023	8.849644	0.763729	0.123992

Table 5.3: Evaluation results with the HetRec LastFM dataset and a random splitting.

In the first one, we selected the MovieLens 1M dataset and we chose a random splitting protocol. For the other parameters, we used the default values of the framework: the test set size is the 20% of the dataset, the length k of the recommended lists is equal to 10, while the rating threshold is equal to 3. The results of this first experiment are reported in Table 5.1.

In the second experiment, we changed the splitting protocol to the timestamp-based one, while we retained all other parameters unmodified. The results are reported in Table 5.2.

Finally, for performing the third experiment, we selected the HetRec LastFM dataset. All other parameters are the same of the first experiment, but the rating threshold, which is now equal to 0. The results of this last experiment are reported in Table 5.3.

As expected, the random recommender always achieves the best coverage, novelty, and diversity, while also obtaining the worst precision, recall, nDCG, and serendipity. On the other hand, the most popular recommender has a very low coverage and novelty, but it also has interesting values of precision and nDCG, especially with the MovieLens dataset. Note its impressive increase in terms of precision obtained by simply changing the splitting protocol.

The measures of serendipity are not exactly zero because this implementation of the most popular recommender suggests a *personalized* list of popular items, avoiding the ones already rated by the same user in the training set.

If we ignore the random suggestions, the Item KNN recommender obtains the best results in terms of coverage, novelty, and diversity in all the experiments. Regarding the metric of precision, we observe interesting results with the User KNN recommender in the first experiment, with the User KNN and the WRMF in the second one, and with the WRMF in the last one. We observe a dramatic decrease in precision of the BPRMF recommender in the LastFM experiment, probably due to the characteristics of the dataset.

In the last experiment, the WRMF algorithm achieves the best values of precision, recall, nDCG, and serendipity. However, it also scores a very low coverage; in contrast, the Item KNN recommender has a fair precision and an interesting coverage. For this reason, it would be useful to compare these algorithms in an online experiment involving human subjects.

The availability of different evaluation metrics was of paramount importance to reach these conclusions, which represent our answer to RQ3.3.

5.5 Conclusion

In this chapter, we have introduced RecLab, an open source framework for evaluating top-k recommender systems in a distributed setting. The main aim of this work is to support the accountability and the reproducibility of the results of

the experiments by permanently storing and publicly displaying their configuration parameters and numerical outcomes.

RecLab is based on a RESTful interaction protocol that enables researchers to evaluate different recommenders created with heterogeneous technologies in a common experimental context and with a comprehensive set of metrics.

The results of each experiment can be easily retrieved and automatically processed using a machine-readable format.

We exploited RecLab for performing three experiments involving all the recommenders at our disposal. We empirically validated the importance of considering different metrics in order to execute a reliable evaluation also in the context of top-k recommender systems and we observed the impact of the configuration parameters on the outcome of the experiments.

The scope of this evaluation framework could be expanded by integrating in it more rating datasets and additional recommendation techniques. We also envision the possibility of enhancing the interaction protocol in order to let the experimenter specify the configuration parameters of each recommender.

Chapter 6

Qualitative Analysis with Rating Datasets Visualization

Being able to correctly interpreting the results obtained during an offline evaluation of different recommender systems is of paramount importance for understanding the quality of the suggested items [47]. However, this task is particularly difficult as it requires knowing several details regarding the evaluation protocol [116]. For this reason, in Chapter 4 and 5, we have formalized two evaluation frameworks to address the problem of reproducibility in recommender systems research.

In the following, we consider the task of characterizing the rating dataset exploited for conducting the experiments, as it has a profound impact on the final result. For example, sparse datasets usually yield to lower evaluation scores with respect to more dense datasets [33]. On the other hand, datasets with many popular items tend to advantage systems that create less diverse suggestions [132], like the most popular baseline. There are also some subtle differences among rating datasets related to the application domain or the collection protocol that could affect the choice of the most appropriate recommender system.

Different metrics have been proposed in literature to summarize the main characteristic of a rating dataset, i.e. sparsity or entropy. However, we argue that such metrics are not sufficient for comparing datasets in a reliable way, as many other facets should be taken into account. For example, it is not possible to understand the rating behaviors of specific groups of users nor the popularity of the most rated items by only looking at some general statistics computed on the whole dataset.

A possible solution to this problem could be represented by data visualization techniques [77]. However, most of the methods available in literature are designed to display the output of a recommendation model and not the original dataset [50, 23]. In contrast, we argue that it is necessary to visually explore a rating dataset even before it is used to train a recommender system, for understanding how the input data will influence the outputs under analysis.

In this chapter, we propose a novel qualitative approach based on data visualization for creating a graphical summary of any collection of user preferences. This method is useful for visually identifying similarities and differences among the available datasets. In fact, we argue that if two datasets result in similar visualizations, the behavior of different recommender systems relying on them will be consistent. Furthermore, we present a Web-based tool, named RS-viz, for easily constructing the proposed visualization and comparing rating datasets in an intuitive way. The software code of RS-viz is freely available on GitHub.[1]

Differently from the plotting capabilities already available in specialized software like Matlab or Scilab, our approach is more general, as it can be applied in a consistent way by different users on any dataset and it can be exploited on many devices without the need of installing specific tools.

More formally, we aim to provide an answer to the following research questions.

RQ4.1 How can data visualization techniques be exploited to create a graphical summary of the main characteristics of a rating dataset?

RQ4.2 To what extent the graphical representation of different rating datasets can be useful to easily identify their similarities and diversities?

The remainder of this chapter is structured as follows. In Section 6.1 we present the approach used to construct the scatter plot and we describe the implementation details of the Web-based tool RS-viz. In Section 6.2, we comment on the outcome of an evaluation campaign designed to validate the proposed method. Finally, in Section 6.3, we provide the conclusions.

6.1 Visualization Approach

In this section, we first describe the algorithm that we devised as an answer to RQ4.1 for creating a scatter plot that represents a rating dataset (Section 6.1.1), then we introduce the implementation details of RS-viz (Section 6.1.2).

6.1.1 Scatter Plot Construction

For visually representing the rating matrix associated with a generic dataset we opted for a 3D scatter plot. The rationale behind this choice is that each point in the visualization could intuitively represent a single rating from the dataset: the value of the x-axis is the identifier of the user, the value of the y-axis is the identifier of the item, while the value of the z-axis is the rating itself, if it is expressed on a numerical scale.

[1] https://github.com/D2KLab/rs-viz

However, it is easy to foresee that this approach cannot handle complex datasets with many preferences, as it requires one point for each rating. If the ratings available are only binary, a traditional scatter plot would suffice.

For these reasons, we decided to create a more compact representation of the rating matrix before visualizing it. In details, we first associated the users and the items with internal numerical identifiers according to their frequency of appearance in the dataset. Therefore, we associated the most rated item with the value of 1, and the second most rated item with the value of 2. The same approach was followed for ordering the identifiers of the users according to the number of ratings that they expressed.

Then, we linearly normalized such identifiers within an interval ranging from 0 to a user provided value, which represents the size of a squared rating matrix in a transformed space. Finally, we binarized the ratings from the original dataset according to a user provided threshold and we counted, for each cell of the transformed matrix, the number of positive preferences associated with that cell.

For example, if the user 40 expressed a preference for the item 360 in a dataset where the number of users is 941, the number of items is 1446, and the number of normalized users and items is equal to 100, that rating would be associated with the cell $(4, 24)$ because $\lfloor 40 \div 941 \times 100 \rfloor = 4$ and $\lfloor 360 \div 1446 \times 100 \rfloor = 24$.

Therefore, the value of the z-axis represents the number of positive ratings associated with a sub-matrix of the original dataset, sorted by item popularity and user activity. In order to enhance the readability of the visualization, we also represented the value of the z-axis using a logarithmic color scale.

As an example of the proposed method, we report in Figure 6.1 and Figure 6.2 the scatter plots obtained from the MovieLens 100K and MovieLens 1M datasets, when the rating threshold is equal to 3, and the number of normalized users and items is equal to 100.

By looking at the values of the z-axis, it is possible to observe in an intuitive way that MovieLens 1M contains a higher number of popular items and of very active users. This conclusion is consistent with the findings of other works that analyze the main characteristics of the MovieLens datasets [33].

6.1.2 Software Implementation

We realized a software implementation of the proposed approach as a Web-based tool, called RS-viz, which is freely available. Our visualization framework has been developed using the JavaScript programming language and it runs entirely in a user's browser. For this reason, it can also be exploited for analyzing private datasets, as no information about them is sent to remote servers.

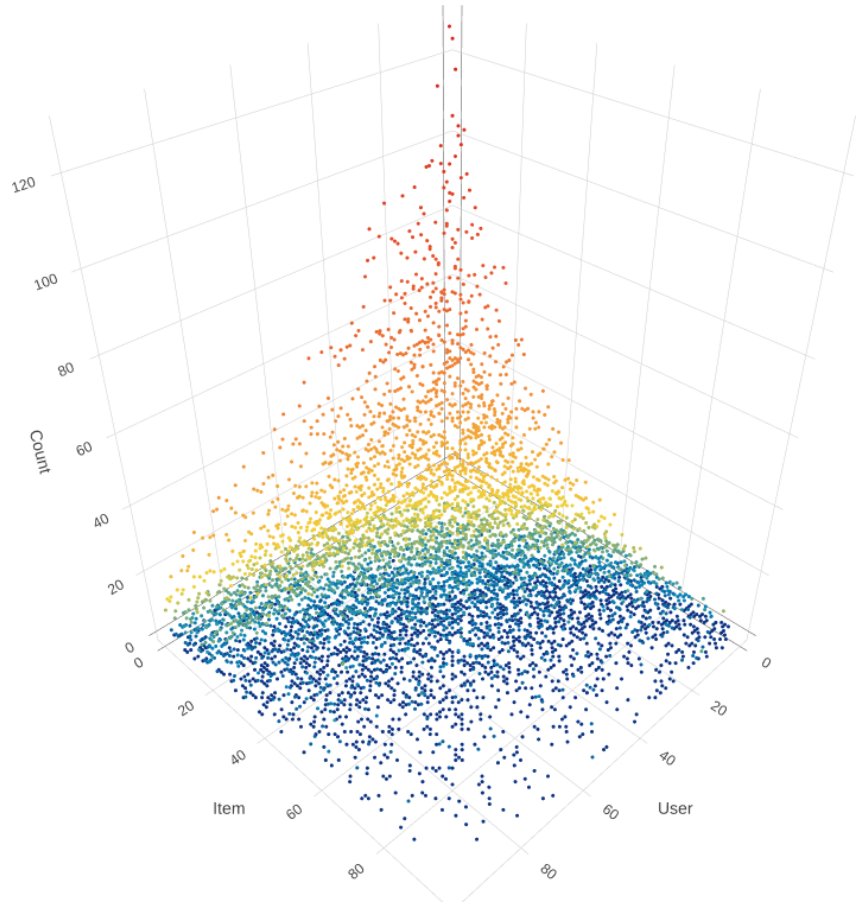


Figure 6.1: A scatter plot representing the MovieLens 100K dataset.

The user needs to visit the Web-page of RS-viz[2] and select one of the built-in datasets or provide her own dataset as a CSV file. Then, she needs to specify the threshold between positive and negative ratings and the number of normalized users and items, which should be selected also considering the rating scale of the input dataset and the desired visualization density. A screenshot of the form containing the configuration parameters of RS-viz is reported in Figure 6.3.

After a few seconds, an interactive 3D scatter plot is constructed on the right side of the page. The user can inspect the plot by rotating the camera and finally save the result as a PNG file.

[2] https://d2klab.github.io/rs-viz/

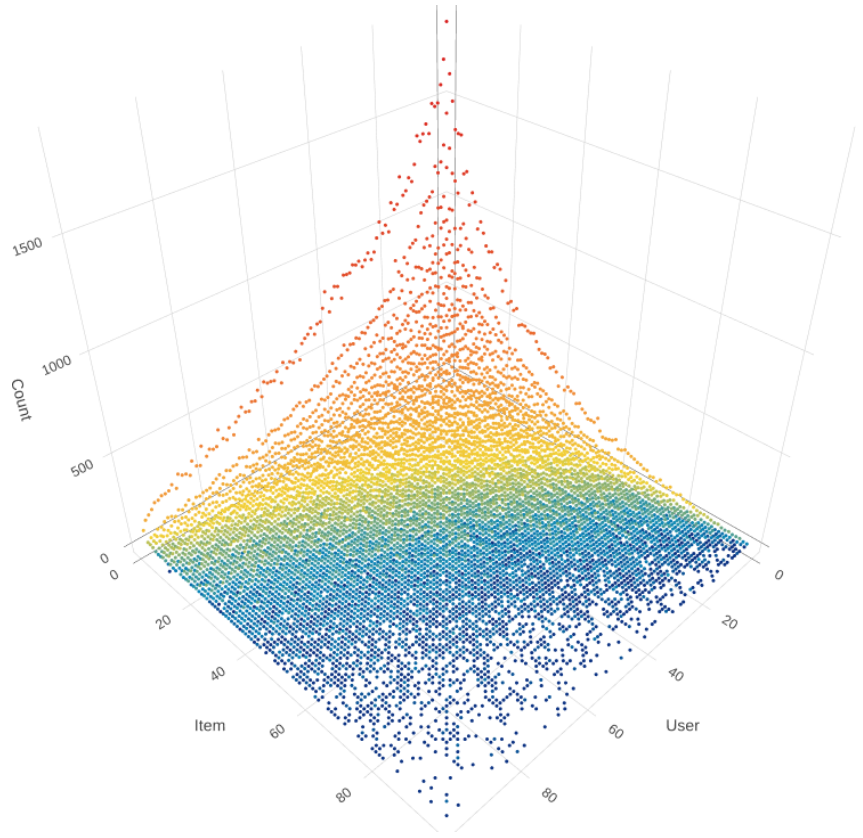


Figure 6.2: A scatter plot representing the MovieLens 1M dataset.

6.2 Evaluation Campaign

In the following, we report the numerical outcomes of an evaluation campaign conducted on the HetRec LastFM dataset using different recommendation approaches with the purpose of understanding if our visualization technique is capable of capturing the different characteristics of a rating dataset and to what extent they influence the recommendation coverage and accuracy.

6.2.1 Experimental Setup

We performed two different experiments with the HetRec LastFM dataset and our evaluation framework RecLab, discussed in Chapter 5.

In the first one, we set the rating threshold equal to 0, while in the second one, we set it equal to 1,000. For the other parameters, we used the default values of the framework: we selected a random splitting protocol, the test set size as the 20% of the dataset, and the length k of the recommended lists equal to 10.

We considered different recommendation approaches, namely the most popular and random baselines and the MyMediaLite [46] implementations of the Item KNN,

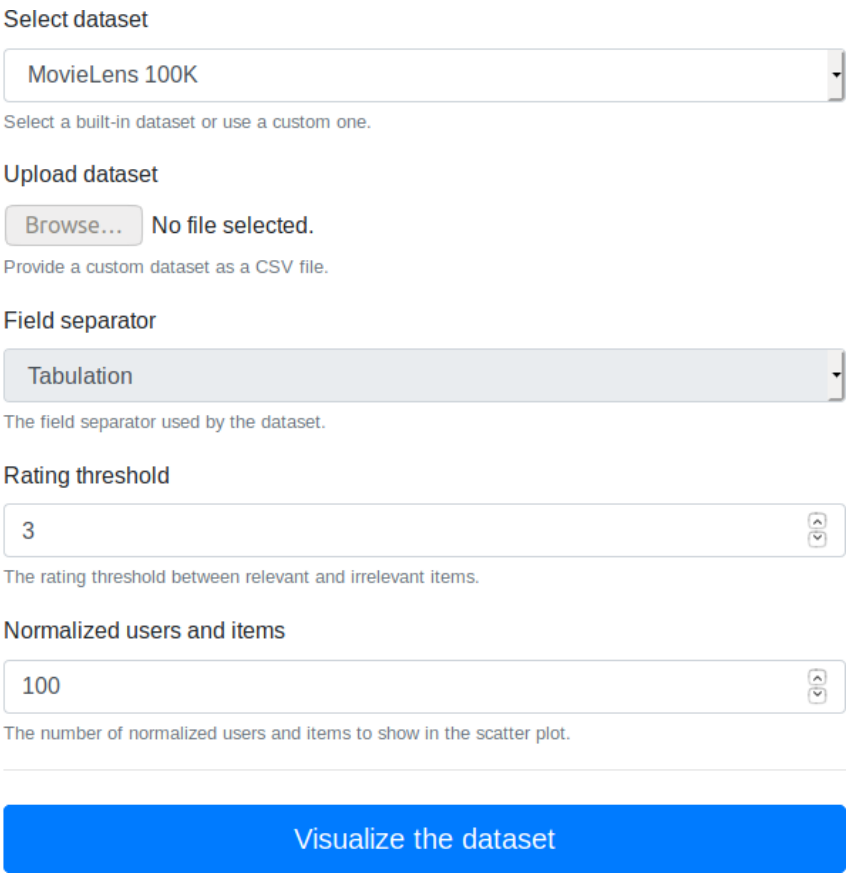


Figure 6.3: The configuration parameters of RS-viz.

User KNN, BPRMF, and WRMF recommender systems using their default settings.

We computed the metrics of coverage, precision, recall, and nDCG. The results of these experiments are reported in Table 6.1. The same datasets obtained from HetRec LastFM by varying the rating threshold were exploited for creating two scatter plots using RS-viz, as displayed in Figure 6.4.

6.2.2 Discussion

From the visualization provided in Figure 6.4a, we can observe that the HetRec LastFM dataset has a very different structure from the one of the MovieLens datasets. In fact, a limited number of items are associated with the preferences of almost all users, as it can be deduced by considering only the ratings expressed for popular items, that is the ones with low identifiers. Please note that such ratings seem not related to the identifier of the user, resulting in a scatter plot that resembles the shape of a half cylinder.

Furthermore, less popular items seem to be liked by less active users. This

Algorithm	Coverage	Precision	Recall	nDCG
Random	0.706679	0.000798	0.000745	0.000858
Most Popular	0.001692	0.071170	0.071480	0.079673
Item KNN	0.235321	0.129362	0.131967	0.145258
User KNN	0.030074	0.157234	0.160353	0.193121
BPRMF	0.022979	0.081277	0.082248	0.094737
WRMF	0.015558	0.159947	0.162332	0.195107

(a) Rating threshold = 0

Algorithm	Coverage	Precision	Recall	nDCG
Random	0.705562	0.000107	0.000622	0.000133
Most Popular	0.001684	0.022122	0.090233	0.027437
Item KNN	0.107233	0.002878	0.013012	0.002686
User KNN	0.049343	0.040672	0.160767	0.055013
BPRMF	0.003756	0.021695	0.088211	0.024366
WRMF	0.012886	0.039606	0.157484	0.053148

(b) Rating threshold = 1,000

Table 6.1: The numerical results of the experimental comparison using the HetRec LastFM dataset.

behavior can be observed by looking at the lower part of Figure 6.4a. Users with a high identifier have rated a more widespread set of items, while users with a low identifier have rated popular items more frequently.

These differences can be easily explained if we consider the collection protocol and the domain of the dataset under analysis. The ratings in the LastFM datasets represent the number of times a user listened to a particular artist: they were collected in an implicit way and their values range from one to tens of thousands.

Also the strange area in the plot with almost no preferences is a direct result of the collection protocol, which relied on the LastFM website to obtain the top artists for a set of users. In fact, the list of artists available in the dataset is limited to 50 items for each user.

If we increase the value of the rating threshold, we can observe that the resulting scatter plot represented in Figure 6.4b is more similar to the ones of the MovieLens datasets, resulting in a very typical long tail distribution with respect to both the items and the users. This outcome is due to the fact that we removed ratings produced by more casual listeners.

From the numerical outcomes of the experiments, we can deduce that the User KNN and WRMF algorithms are the most appropriate ones with both the different

rating thresholds. In general, all the recommenders available perform worse with a higher threshold. In fact, from the visualizations it is clear that the number of available preferences is much lower with respect to the MovieLens 100K dataset, as the scatter plot represented in Figure 6.4b is sparser than the one available in Figure 6.1. Because user preferences are more limited in number and fragmented, the task of any recommender system is necessarily harder.

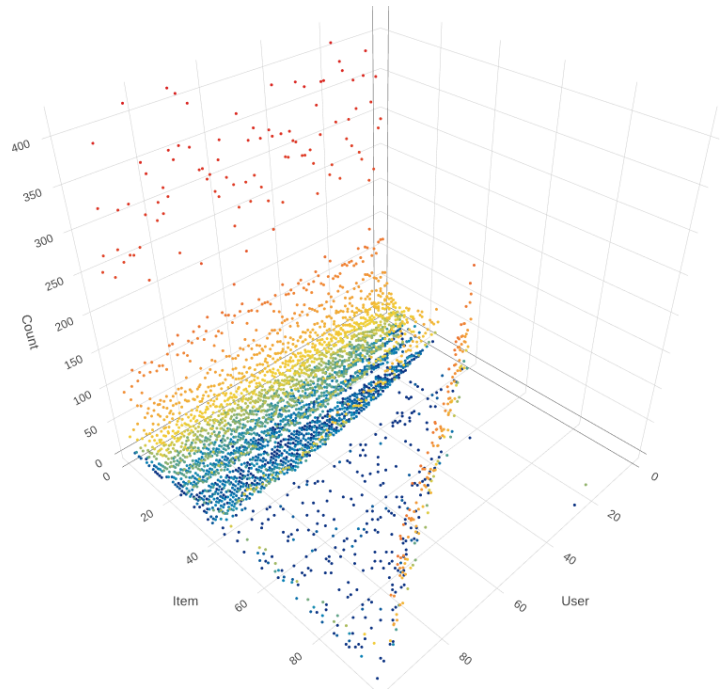
Interestingly, the Item KNN, differently from the User KNN, experienced a dramatic drop in all the metrics considered. This result may have been caused by the fact that a very low number of users is available for each item of the dataset. Also this characteristic can be observed from the generated scatter plot by looking at the lower part of Figure 6.4b. The white horizontal stripes denote groups of items that have been rated by only a few very active users.

These experimental results support the validity of our visualization approach and they represent our answer to RQ4.2.

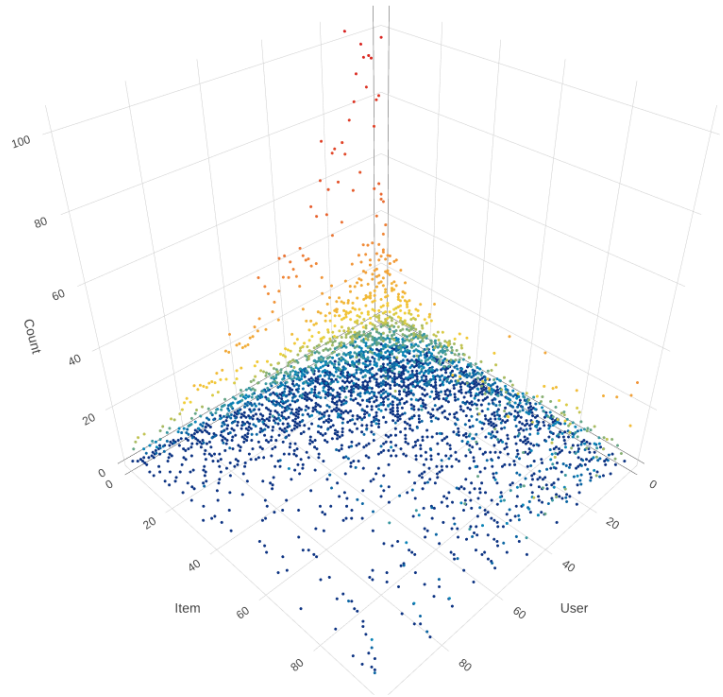
6.3 Conclusion

In this chapter, we proposed a method for creating graphical summaries of any rating dataset for the purpose of enabling researchers and practitioners to better interpret the results of an offline evaluation campaign. Furthermore, we introduced RS-viz, a Web-based tool capable of creating an interactive 3D scatter plot according to the aforementioned approach starting from a user provided CSV dataset or a built-in collection of ratings.

We validated the capabilities of such visualizations to reveal useful information by comparing the graphical representations of the HetRec LastFM dataset constructed with different rating thresholds with the numerical outcomes of two offline experiments involving various recommendation techniques.



(a) Rating threshold = 0



(b) Rating threshold = 1,000

Figure 6.4: The scatter plots obtained using the HetRec LastFM dataset with different rating thresholds.

Chapter 7

Generation and Evaluation of Synthetic Datasets

It is necessary to rely on a collection of user preferences obtained in a particular domain to perform an offline experiment. For example, the MovieLens datasets represent a popular choice for conducting an offline evaluation in the field of movie recommender systems [57]. Nevertheless, the number and the variety of publicly available rating datasets is often limited, especially in less mainstream domains [127]. It is possible to identify different causes for this problem. For example, the companies capable of collecting rating datasets are usually reluctant to share them, because of the fear of violating the privacy of their users or of exposing commercially sensible data to their competitors. On the other hand, researchers often do not have the resources for obtaining a sufficient number of ratings that are worth to be publicly released, as discussed in Section 3.2.8.

Because of the shortage of public datasets, practitioners have started to rely on synthetic ratings in order to conduct their offline experiments [137]. An obvious advantage of such an approach is that it enables the creation of rating datasets with an arbitrary number of users and items at a limited cost of dataset acquisition. However, the results obtained from such experiments may be questionable, as the generated datasets are usually not capable of capturing the characteristics of a particular domain of interest [88]. For example, different generative approaches only rely on descriptive statistics, like mean and standard deviation, and, for this reason, they fail to mimic the individual behavior of a user.

In this chapter, we propose a novel approach for automatically generating synthetic datasets with a configurable number of users leveraging on a reference dataset that is used as the seed of the process and that encodes the peculiarities of a domain of interest. Such a generative method can be exploited to create different rating datasets containing users that exhibit behaviors similar to the ones available in the reference dataset. However, the synthetic users do not have a direct relation with the real users and, therefore, no private or commercially sensible information

is leaked. At the same time, because the number of synthetic users is configurable, the generated dataset can be exploited to conduct scalability tests in a realistic way and to train recommendation algorithms using reinforcement learning approaches.

More formally, we aim to provide an answer to the following research questions.

RQ4.3 What is the impact of using a synthetic dataset instead of a real one on the results of an offline experiment in the context of recommender systems?

RQ4.4 Can a generative approach be exploited to create a synthetic dataset that exhibits properties similar enough to the ones of a real dataset?

RQ4.5 To what extent this method can be consistently applied to datasets from different domains and of different sizes?

The remainder of this chapter is structured as follows. In Section 7.1 we introduce the generative approach for creating synthetic datasets, while in Section 7.2 we describe the experimental setup designed to validate it. We present and discuss the results in Section 7.3 and, in Section 7.4, we provide the conclusions.

7.1 Dataset Generation

Our approach for generating synthetic datasets starting from a reference dataset consists of two steps. In the first one, it is necessary to analyze an existing collection of user preferences in order to obtain an accurate representation of the domain of interest. Then, in the second one, it is possible to exploit such a representation for creating different generated datasets.

We argue that only relying on a few statistical distributions computed empirically at a global level from an existing dataset or specified by a researcher is not sufficient to realistically simulate the individual tastes of human beings [88]. Such methods would lead to the creation of datasets with users having no individual preferences, thus making the task of any recommender system nearly impossible.

For this reason, we included a preliminary clustering phase as part of the first step in order to group the users in a fixed number of communities. The individual rating behaviors, represented by different statistical distributions, are learned for each community and then exploited during the sampling phase.

For simplicity, we assume that each user can only express positive preferences about the items available in the system. However, this approach can also be exploited to simulate datasets with ratings expressed on a more complex scale by repeating these steps for each rating value and then by merging the results.

In the following, we detail the user clustering and distribution learning process (Section 7.1.1) and the rating sampling algorithm (Section 7.1.2).

7.1.1 User Clustering and Distribution Learning

We represent each user $v \in \mathcal{U}$ from the reference dataset as a vector with length equal to the number of items $|\mathcal{I}|$. The component $\hat{v}_i$ of such a vector is equal to 1 if the user v expressed a positive rating ρ about the i-th item of the catalog, otherwise it is equal to 0.

Given this data structure, we decided to apply the K-means clustering algorithm [58] to group together users who liked a similar set of items in K different clusters. In the following, we define $\mathcal{C}$ as the set of clusters, therefore $|\mathcal{C}| = K$. The value of K needs to be empirically selected by the experimenter because, in general, it depends on the characteristics of the reference dataset.

Every cluster identifies a different community of users. For generating a dataset similar to the reference one, it is necessary to know how many users belong to each community and what are the item preferences associated with them. More in detail, we create the following empirical distributions from the reference ratings:

- P^C, how users are distributed in K clusters;
- P_k^U, how ratings are distributed in $|\mathcal{U}|$ users for each cluster;
- P_k^I, how ratings are distributed in $|\mathcal{I}|$ items for each cluster.

Note that only the first distribution is global, while the second and the third ones are associated with a cluster.

The distribution P^C represents the probability of assigning a user to a certain cluster and it is computed by counting the number of users per cluster. The distribution P_k^U represents the probability of finding a certain number of ratings per user in the cluster k and it is computed by counting the number of ratings per user. Finally, the distribution P_k^I represents the probability of finding a certain number of ratings per item in the cluster k and it is computed by counting the number of ratings per item.

The user clustering and distribution learning process is formalized in Algorithm 7.3. Its output is represented by the previously mentioned distributions.

7.1.2 Rating Sampling

Starting from the empirical distributions obtained from Algorithm 7.3, it is possible to generate a synthetic dataset by applying to them a sampling function σ. In the following, we assume that σ is the weighted random sampling function.

As previously mentioned, the experimenter can select the number of users available in the generated dataset. This value, called U, is an input of the rating sampling algorithm, together with the probability distributions. The synthetic dataset can also have the same number of users available in the reference dataset, that is $U = |\mathcal{U}|$, in order to create a more realistic dataset.

Algorithm 7.3 User clustering and distribution learning, given a reference dataset and the number of clusters.

Require: $\mathcal{U} \neq \{\emptyset\} \wedge K > 0 \wedge K \leq |\mathcal{U}|$
1: $\mathcal{C} \leftarrow$ K-means$(\mathcal{U}, K)$
2: $P^C \leftarrow P(\upsilon \in \mathcal{C}_k)$
3: **for all** $k \in \{1, \ldots, K\}$ **do**
4: $\quad P_k^U \leftarrow P(\rho_\upsilon | \upsilon \in \mathcal{C}_k)$
5: $\quad P_k^I \leftarrow P(\rho_\iota | \iota \in \mathcal{I}_\upsilon \wedge \upsilon \in \mathcal{C}_k)$
6: **end for**
7: **return** $P^C,\ P_k^U,\ P_k^I$

Firstly, each generated user u is assigned to a cluster k from the reference dataset, according to the distribution of users per cluster. Then, the number of ratings I for that user is selected considering the distribution of ratings per user in the cluster k. Finally, I items are sampled without replacement ($\hat{\sigma}$) from the distribution of ratings per item in the cluster k. Thus, the number of user ratings and her liked items are associated with a particular community of users.

The rating sampling procedure is formalized in Algorithm 7.4.

Algorithm 7.4 Rating sampling, given the required number of users and the distributions computed in Algorithm 7.3.

Require: $U > 0, P^C, P_k^U, P_k^I$
1: $\mathcal{R} \leftarrow \{\emptyset\}$
2: **for all** $u \in \{1, \ldots, U\}$ **do**
3: $\quad k \leftarrow \sigma(P^C)$
4: $\quad I \leftarrow \sigma(P_k^U)$
5: $\quad$ **for all** $i \in \{1, \ldots, I\}$ **do**
6: $\quad\quad \rho_{u,i} \leftarrow \hat{\sigma}(P_k^I)$
7: $\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup \{\rho_{u,i}\}$
8: $\quad$ **end for**
9: **end for**
10: **return** $\mathcal{R}$

7.2 Experimental Setup

We compared the results obtained from the evaluation of different recommenders conducted on popular datasets typically exploited in literature with the ones computed in the same experimental conditions using various collections of synthetic preferences generated starting from them using multiple techniques.

In fact, we claim that a synthetic dataset can be successfully used during an evaluation campaign if the behavior of the recommender systems under analysis is similar to one that it would be possible to observe with the reference dataset. Thus, almost all the possible pairs of recommenders should exhibit the same relation of order for a given dimension and lead to similar conclusions.

Furthermore, we investigated what is the impact of the parameter K on the results of the evaluation, in order to understand how to empirically select the most appropriate value for it.

In our experiments, we utilized Random, Most Popular, User KNN, BPRMF, and WRMF recommendation algorithms and the metrics of precision, recall, and nDCG as defined in the evaluation framework RecLab discussed in Chapter 5. Regarding the user preferences, we exploited the binarized versions of the MovieLens 100K, MovieLens 1M, and LastFM [22] datasets. We considered as positive all ratings with a value higher than 3 for MovieLens and than 0 for LastFM. We relied on the default values of the evaluation framework for all other experimental parameters: we followed a random splitting protocol with a test set size equal to the 20% of all available ratings and we recommended 10 items for each test user.

From the aforementioned reference datasets we generated their synthetic versions exploiting the procedure described in Section 7.1. We considered U equal to the number of users originally available, in order to compare datasets of similar size. Furthermore, we also created three baseline synthetic collections with the same number of ratings by not applying the user clustering phase. All the users of such baselines exhibit the same rating behavior, similarly to the approach described in [25]. In Table 7.1, we report different statistics regarding the baseline, generated, and reference datasets.

Dataset	Version	Users	Items	Ratings
MovieLens 100K	Baseline	942	1,374	55,375
MovieLens 100K	Generated	942	1,332	53,915
MovieLens 100K	Reference	942	1,447	55,375
MovieLens 1M	Baseline	6,038	3,463	575,281
MovieLens 1M	Generated	6,038	3,457	584,101
MovieLens 1M	Reference	6,038	3,533	575,281
LastFM	Baseline	1,888	13,342	92,834
LastFM	Generated	1,892	13,442	92,510
LastFM	Reference	1,892	17,632	92,834

Table 7.1: The total number of users, items, and ratings available in the datasets.

7.3 Results

In this section, we first discuss the impact of the number of user communities on the evaluation results, then we present a comparison between exploiting the synthetic and the reference datasets.

7.3.1 Number of User Communities

Dataset	Most Popular	User KNN	BPRMF	WRMF
K = 5	0.088449	0.099890	0.078768	0.091749
K = 10	0.095793	0.124595	0.102805	0.111974
K = 50	0.098378	0.133946	0.103243	0.133838
K = 100	0.102415	0.150494	0.115587	0.149945
K = 200	0.099672	0.154158	0.122538	0.164114

Table 7.2: The values of precision obtained with the synthetic versions of the MovieLens 100K dataset by varying K.

For studying what is the impact of the value K on the results of an evaluation conducted with a synthetic dataset, we computed the measure of precision on different synthetic versions of the MovieLens 100K dataset created with $K = \{5,\ 10,\ 50,\ 100,\ 200\}$. We report the numerical outcomes of this experiment in Table 7.2.

We also observed that it is possible to obtain similar results by considering other datasets and metrics. As expected, the values of precision for all the algorithms but the Random and Most Popular approaches improve by increasing the number of available clusters. However, this relationship is not linear, as doubling its value from 100 to 200 only slightly improves the results.

We empirically observed that reasonable values for K could be 100 or 200. In Section 7.3.2, we will assume that $K = 200$.

Therefore, we can provide an answer to RQ4.3 by observing that the impact of using a synthetic dataset in an evaluation campaign can be mitigated if we are able to simulate a sufficient number of heterogeneous user communities.

7.3.2 Synthetic and Reference Datasets

As anticipated in Section 7.2, we compared the evaluation results obtained when relying on the reference dataset and two synthetic datasets created with different approaches. We repeated this experiment with datasets of different sizes and from different domains in order to assess the generalizability of the results.

Algorithm	Precision	Recall	nDCG
Random	0.009416	0.008877	0.009841
Most Popular	0.060065	0.053209	0.064384
User KNN	0.055952	0.050587	0.058744
BPRMF	0.045346	0.033628	0.048740
WRMF	0.047078	0.042876	0.048104

(a) Baseline dataset

Algorithm	Precision	Recall	nDCG
Random	0.009847	0.008977	0.010022
Most Popular	0.099672	0.083875	0.110229
User KNN	0.154158	0.135917	0.169499
BPRMF	0.122538	0.106186	0.129742
WRMF	0.164114	0.144272	0.173916

(b) Generated dataset

Algorithm	Precision	Recall	nDCG
Random	0.007743	0.006300	0.008183
Most Popular	0.112759	0.102804	0.130632
User KNN	0.205234	0.221684	0.233362
BPRMF	0.182770	0.186838	0.198869
WRMF	0.221592	0.233235	0.250386

(c) Reference dataset

Table 7.3: The results obtained with the baseline, generated, and reference versions of MovieLens 100K.

The results obtained with MovieLens 100K, MovieLens 1M, and LastFM are available in Table 7.3, Table 7.4, and Table 7.5 respectively.

We observe that in all experiments and for almost all the possible pairs of recommenders the relative order of the measures is the same between the generated and the reference datasets.

As expected, their values are lower when exploiting the synthetic ratings, as they do not represent real preferences. Nevertheless, they are still useful to identify the most promising recommendation techniques in a certain domain, while the results obtained with the baseline datasets cannot be exploited for such a purpose.

With respect to RQ4.4, we can conclude that a generative approach capable of replicating the behaviors of different groups of users can be used for creating

Algorithm	Precision	Recall	nDCG
Random	0.004989	0.002733	0.004803
Most Popular	0.066483	0.037723	0.068991
User KNN	0.064708	0.035976	0.066573
BPRMF	0.057459	0.027145	0.059226
WRMF	0.053022	0.027763	0.055035

(a) Baseline dataset

Algorithm	Precision	Recall	nDCG
Random	0.005712	0.002678	0.005580
Most Popular	0.101570	0.061565	0.107356
User KNN	0.129113	0.078716	0.136779
BPRMF	0.106948	0.057729	0.111058
WRMF	0.133723	0.080045	0.140670

(b) Generated dataset

Algorithm	Precision	Recall	nDCG
Random	0.005589	0.002862	0.005657
Most Popular	0.131982	0.082978	0.142782
User KNN	0.232082	0.172290	0.262018
BPRMF	0.199633	0.136378	0.218727
WRMF	0.227878	0.154425	0.252999

(c) Reference dataset

Table 7.4: The results obtained with the baseline, generated, and reference versions of MovieLens 1M.

realistic datasets. We also discovered, as an answer to RQ4.5, that our approach can be potentially applied to datasets from different domains and of different sizes.

7.4 Conclusion

In this chapter, we have discussed a method for generating synthetic datasets with an arbitrary number of users starting from existing collections of preferences. Differently from the approaches already available in literature, we propose to first model user communities in order to generate more realistic ratings that can be successfully exploited during an evaluation campaign.

Algorithm	Precision	Recall	nDCG
Random	0.000691	0.000656	0.000869
Most Popular	0.046281	0.046513	0.048636
User KNN	0.042614	0.043088	0.044308
BPRMF	0.039957	0.040543	0.041804
WRMF	0.032731	0.032974	0.033975

(a) Baseline dataset

Algorithm	Precision	Recall	nDCG
Random	0.000532	0.000520	0.000548
Most Popular	0.052844	0.054597	0.056285
User KNN	0.101435	0.104549	0.113433
BPRMF	0.062307	0.064720	0.066835
WRMF	0.090324	0.092860	0.099598

(b) Generated dataset

Algorithm	Precision	Recall	nDCG
Random	0.000797	0.000825	0.000791
Most Popular	0.067906	0.068970	0.075406
User KNN	0.156057	0.160451	0.189487
BPRMF	0.075877	0.077336	0.087066
WRMF	0.160202	0.164468	0.193937

(c) Reference dataset

Table 7.5: The results obtained with the baseline, generated, and reference versions of LastFM.

We empirically verified that the outcome of an offline comparison among different recommender systems conducted exploiting the generated datasets is consistent with the results obtained when using the reference datasets, provided that a sufficient number of user clusters is selected. This finding could encourage private companies to publicly release synthetic datasets created from internally available data without the fear of violating the privacy of their users or of exposing commercially sensible information.

Chapter 8

First Use Case: Semantic Review Recommender

During the last decade, the Web has evolved from an information space to share textual documents into a medium to distribute structured data. Linked Data[1] is a set of best practices for publishing and interlinking data on the Web and it is the base of the Web of Data, an interconnected global knowledge graph. Because of the increased amount of machine-readable knowledge freely available on the Web, there is a high interest in investigating how such information can be used to improve recommender systems [44], as reviewed in Section 2.1.4.

Currently, most recommender systems exploit ratings to infer user preferences, although the growing popularity of social and e-commerce websites has encouraged users to write reviews. These reviews enable recommender systems to represent the multi-faceted nature of users' opinions and build a fine-grained preference model, which cannot be obtained from overall ratings [27]. Additionally, as discussed in Section 2.1.3, recommender systems may take advantage of reviews because they are harder to fake than ratings, are richer of information, and users may struggle to express their preference as ratings. Some studies have also documented the positive influence of product reviews on the decision processes of new users [26, 75].

In this chapter, we address the issue of mining reviews and show how the extracted information, combined with Linked Data, can be exploited in recommendation tasks. On one side Linked Data can provide a rich content-based representation of the items to be recommended since they include interesting features. For example, movies represented in DBpedia[2] contain basic information such as cast and director, but also some unexpected relations, such as the fact that both *Braveheart* and *Saving Private Ryan* won the Best Sound Editing Academy Award. On the

[1] http://linkeddata.org

[2] http://dbpedia.org

other side, reviews may reveal additional connections among items. For instance, various reviews of *Interstellar* mention Stanley Kubrick, although in DBpedia there is not a direct link between these two resources.

Therefore, we propose a new recommendation approach that semantically annotates reviews to extract useful information from them. The annotated entities and the knowledge freely available in the Web of Data are then combined to discover additional resources and generate recommendations. Our method can exploit any dataset available in the Web of Data to provide recommendations, although we rely on DBpedia and Wikidata[3] in our implementation.

We conducted an offline study to find the best configuration of our technique for these two datasets and comparatively evaluate our approach against a Linked Data-based and some more traditional algorithms based on ratings. We performed our study in the movie, book, and music domains, and the evaluation took into account different properties of recommender systems, that is prediction accuracy (both in terms of ratings and ranking), diversity, and novelty using a multicriteria approach. In fact, as discussed in Section 2.2.2, not only accuracy is important: recommendations that are too obvious or already known to users may not satisfy them, although they match their taste. The results showed that our method achieved the highest diversity, provided a better accuracy than the approach based on Linked Data, and increased the novelty of recommendations with respect to collaborative filtering techniques.

The remainder of this chapter is organized as follows. In Section 8.1 we present our approach, while, in Section 8.2, we describe the evaluation method. Then, in Section 8.3, we show the obtained results and in Section 8.4 we discuss them. Finally, in Section 8.5, we provide the conclusions.

8.1 Approach

The architecture of SemRevRec is depicted in Figure 8.1. The system consists of two main modules that are highlighted with different colors: semantic annotation and discovery, and recommendation. The former is responsible for feeding the recommender system with semantically annotated entities and Linked Data through the knowledge base, while the latter provides recommendations to users. Every time a new review is submitted, the system executes the semantic annotation and discovery steps and possibly adds new entities, while the recommendation process can start when the user provides an initial item. The recommendation module works online, while the semantic annotation and discovery are done offline. Initially, some reviews are annotated and the resulting entities are used to discover additional entities through Linked Data. Each of these two modules is made up

[3] https://www.wikidata.org

of the illustrated submodules, which are responsible for specific steps of the whole process: annotation, discovery, generation of recommendations, and their ranking. The storage of entities is not a step, but the corresponding database is a transversal submodule used by all the others.

SemRevRec deals with the annotated or discovered entities and the items to recommend. We consider the items as a particular type of entities since SemRevRec suggests items that may be annotated or discovered entities. An item may not appear as an entity in the system, for instance a movie that was reviewed but never annotated or discovered. However, this does not mean that an entity corresponding to such a movie does not exist in the considered knowledge base. Semantic annotation and discovery steps are explained in Section 8.1.1, while the recommendation process is presented in Section 8.1.2.

Although our approach is not bound to a particular domain or knowledge base available in the Web of Data, in our implementation we focus on movies, books, and music, while we rely on DBpedia and Wikidata to identify possible differences between these two knowledge bases. We chose them for annotation and discovery because they are two of the main datasets in the Web of Data and they have a vast amount of resources that belongs to a variety of domains. We used reviews from IMDb[4] for movies, LibraryThing[5] for books, and Amazon[6] for music.

8.1.1 Semantic Annotation and Discovery

Semantic annotation is the process of annotating textual or multimedia contents with semantic tags to add information about their meaning [115]. In written text, this can be done by associating a URI to the recognized entities. We considered two popular semantic annotators that rely on Wikipedia: AIDA [65] and DBpedia Spotlight [35]. They are both capable of disambiguating entities according to the surrounding context: this is useful because users frequently write acronyms and abbreviations. We finally selected AIDA because it is more accurate according to an independent comparison [45].

The module of semantic annotation and discovery analyzes the text of the reviews and stores the identified entities in a relational database. The URI of each annotated entity is associated with the URI of the reviewed item and with the occurrence of that entity in all the reviews of that item. In fact, the same entity may appear again in reviews regarding another item. AIDA is capable of identifying and disambiguating entities mentioned in the review considering, by default, the

[4] http://www.imdb.com

[5] https://www.librarything.com

[6] https://www.amazon.com

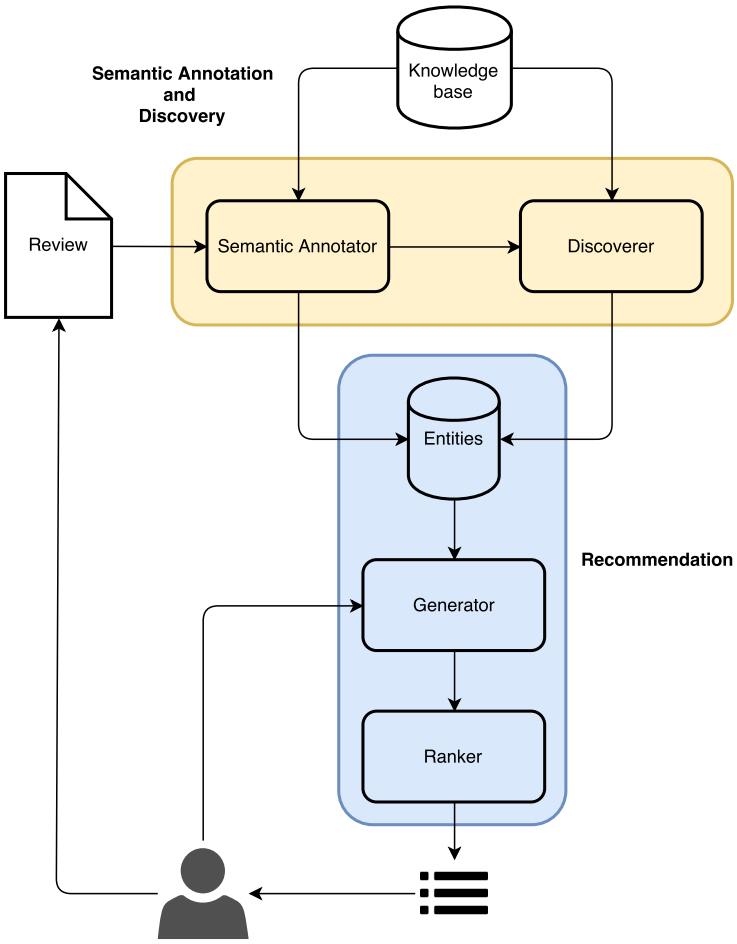


Figure 8.1: The system architecture of SemRevRec.

ones available in YAGO.[7]

The YAGO resources are mapped with the equivalent ones available in DBpedia exploiting the similar structure of the URIs. For example, `yago-res:The_Matrix` corresponds to `dbr:The_Matrix` because their URIs where both generated starting from the title of the same Wikipedia article. In contrast, the mapping between DBpedia and Wikidata relies on the `owl:sameAs` predicate available in DBpedia. If the same entity corresponds to more than one in the other knowledge base, it is ignored in order to avoid probable inconsistencies. The same holds if there is no `owl:sameAs` property. In principle, it is also possible to perform the semantic annotation phase relying on a custom knowledge base, but AIDA is provided with a precomputed database that includes all the necessary information for annotating using YAGO. In our case, since DBpedia and Wikidata are both well interlinked

[7] http://www.yago-knowledge.org

with YAGO, it was less time consuming computing the mapping rather than the information needed by the annotator.

Finally, the types of each entity are obtained from the target knowledge base, optionally considering only a subset of them (for example, only the DBpedia ontology types, such as `dbo:Film`). This is done in order to minimize the amount of information retrieved and to reduce the time required for this operation. The types are stored locally because they are not expected to change often and reading them from a database is more efficient than querying the original knowledge base.

Semantic annotation allows SemRevRec to exploit Linked Data for retrieving additional entities. This is possible because the annotated entities are also resources in the Web of Data. Thus, the discoverer can find resources that are related to the annotated entities in order to enable our system to recommend more items. Reviews are a source of non-trivial relations: for example, in a movie recommendation scenario, a user can mention a movie that reminds her the reviewed one because of the colors, the setting, or the atmosphere, and these features are hardly available as Linked Data. At the same time, Linked Data can enrich information coming from users. For instance, they enable the discoverer to obtain other movies in which an actor mentioned in a review played. The discovery can take into account various properties, from more traditional ones, such as the genre, the director, or the actors, to more unexpected ones, such as other movies shot in the same place.

Given the annotated entities, the discoverer retrieves from the knowledge base other relevant entities through SPARQL queries. It relies on some properties that can be configured and depend on the domain and on the dataset considered. The discovery is not bound to a particular knowledge base or domain. On the contrary, this approach is fairly general since it relies only on RDF and SPARQL. In our implementation, we considered DBpedia and Wikidata, and we focused on movie, book, and music recommendations. Table 8.1 summarizes the properties that we selected for discovering further items to recommend starting from the entities available in the reviews.

Domain	DBpedia	Wikidata
Movie	`dbo:starring`	`wdt:P161`
Movie	`dbo:director`	`wdt:P57`
Book	`dbo:author`	`wdt:P50`
Music	`dbo:artist`	`wdt:P175`
Music	`dbo:writer`	`wdt:P676`

Table 8.1: The properties considered for the discovery phase.

More specifically, the discoverer reads the annotated entities stored during the semantic annotation phase. The discoverer is able to obtain all the resources that have the given entities as an object of the selected properties. For example, in the

movie domain, we selected `dbo:starring` and `dbo:director` in the case of DBpedia because most of the annotated properties, when not movies, were actors and directors. This allows the system to discover other movies from the same director or actor named in a given review. Sometimes directors or actors not involved in the movie were also mentioned for comparison. The discoverer can retrieve other movies from these entities that are relevant for the user who wrote the review, thus can also be of interest for other users. Similarly to movies, we selected `dbo:author` for books as well as `dbo:artist` and `dbo:writer` for music because most of the annotated entities were authors, artists or writers when not books and songs, respectively. It is possible to exploit both direct and inverse properties.

The discoverer stores the discovered entities in a relational database for efficiency reasons. The URI of each discovered entity is associated with the URI of the annotated entity through which it was discovered, and, optionally, with the LDSD measure [104] between them. This measure is inversely proportional to the number of links between two resources: more links result in a lower distance. Each discovered entity may be found through more than a single annotated entity. The LDSD can be exploited in the ranking phase, which is described in Section 8.1.3. However, since its computation is expensive due to the various SPARQL queries involved, it may be optionally skipped to speed up the discovery step. Obviously, in this case, the LDSD measure does not contribute to the ranking.

8.1.2 Recommendation

The recommendation process consists of two main steps: the generation of the candidate recommendations and their ranking. Given an initial item, SemRevRec retrieves all the entities that are related to the initial item and then ranks them.

Firstly, the system selects the annotated entities that were mentioned in the reviews of the initial item. Afterwards, it obtains the entities that mention the initial item, that is entities whose reviews generated an annotated entity that corresponds to the initial item. For example, if the initial item is *Interstellar* and a review of *2001: A Space Odyssey* mention *Interstellar*, then *2001: A Space Odyssey* is considered as a candidate recommendation.

Secondly, SemRevRec optionally retrieves the discovered entities. They may include entities discovered through the initial item. For instance, if the initial item is *Interstellar* and *The Dark Knight* was previously discovered because both these movies have been directed by Christopher Nolan, *The Dark Knight* is selected. The same holds if *Interstellar* was discovered from *The Dark Knight*, that is Christopher Nolan was annotated in the reviews of the latter. Similarly, the entities discovered through other entities that were annotated in the reviews of the initial item are relevant. For example, if *Interstellar* is the initial item, Stanley Kubrick was annotated in one of its reviews, and *2001: A Space Odyssey* was discovered through Stanley Kubrick, then *2001: A Space Odyssey* is a candidate recommendation.

It is possible to configure the generator to include in the candidate recommendations the discovered entities or not. It is also possible to specify the minimum occurrence required for entities to be included in the candidate recommendation set, which is expressed as a percentage of the maximum occurrence of entities in the reviews of the item considered.

8.1.3 Ranking Functions

Finally, SemRevRec ranks the candidate recommendations. We defined three different ranking functions. The first one is presented in Equation 8.1 and takes into account only the occurrence $occur(i)$ of the entities available in the reviews. $occur(i)$ is equal to the number of reviews of an initial item i_{in} where an entity i is annotated plus the number of reviews of i where i_{in} is annotated (if any). However, the entity i can be annotated or discovered. For the latter, the occurrence of the entity through which it was discovered is used. The α coefficient is 1 if i is an annotated entity. Otherwise, it can be configured to a custom value (the default is 0.5) to weight the contribution of a discovered entity to the ranking. To obtain a value between 0 and 1, R1 is normalized to the maximum occurrence of entities j that belong to the candidate recommendation set CR.

$$R1(i) = \frac{\alpha \cdot occur(i, i_{in})}{max_{j \in CR}(occur(j, i_{in}))} \tag{8.1}$$

The second ranking function (Equation 8.2) also considers the LDSD measure between each discovered entity and the entity through which it was discovered. This avoids assigning the same value to all the entities discovered through the same annotated entity as R1 does. As for R1, the entity i can be annotated or discovered. The β coefficient is 1 if i is an annotated entity, 0.5 otherwise. The γ coefficient is 0.5 for discovered entities, 0 otherwise. In this way, R2 returns a number between 0 and 1, which is equal to R1 for the annotated entities, while, for the discovered entities, it is the average of R1 and $LDSD(i, i_o)$, where i_o is the entity through which i was discovered.

$$R2(i) = \beta \cdot R1(i) + \gamma \cdot (1 - LDSD(i, i_o)) \tag{8.2}$$

The third ranking function (Equation 8.3) considers the LDSD measure between an entity i and the initial item i_{in}. The coefficients η and κ can be set to custom values and they allow the ranker to weight differently the contribution of the occurrence in the review (given by R2) and Linked Data (through the LDSD).

$$R3(i) = \eta \cdot R2(i) + \kappa \cdot (1 - LDSD(i, i_{in})) \tag{8.3}$$

LDSD measures between discovered entities and the entities through which they were discovered need to be precomputed at discovery time (see Section 8.1.1) to

enable SemRevRec to exploit R2, LDSD measures between entities in CR and the initial item need to be computed while ranking.

8.2 Evaluation Procedure

We evaluated the performance of SemRevRec with two offline experiments conducted in the movie, book, and music domains. The purpose of the first experiment is to understand the impact of the ranking function, the discovery, the occurrence threshold, and the coefficients of R3. Furthermore, we performed the first experiment two times, first relying on DBpedia and then on Wikidata, to assess the effect of the knowledge base on the quality of the recommended items. The aim of the second experiment is to compare our proposal with traditional recommendation techniques that rely on ratings and a recommender system based on Linked Data.

For conducting both experiments, we obtained from IMDb, LibraryThing, and Amazon the user reviews regarding all the items included in the MovieLens 1M,[8] the LibraryThing[9] and the HetRec 2011 LastFM[10] datasets of user ratings.

The items of such rating datasets were mapped with the corresponding entities available in DBpedia relying on the work of Di Noia et al. [39]. Furthermore, their equivalent entities in Wikidata were obtained from DBpedia itself, as described in Section 8.1.1. For the purpose of retrieving the user reviews, Wikidata was exploited in order to discover the IMDb identifiers of the movies available in the MovieLens 1M dataset. On the contrary, the LibraryThing dataset already contained the references useful for obtaining the reviews. Regarding the musical artists present in the HetRec 2011 LastFM dataset, we relied on the search feature of Amazon for identifying their most reviewed musical work.

	Movie	Book	Music
Users	6,040	7,279	1,892
Items	3,706	37,232	17,632
Ratings	1,000,209	2,056,487	92,834
Reviews	559,858	363,791	669,978
Distinct entities	107,468	77,120	70,762
Total entities	574,435	303,705	296,777

Table 8.2: Statistics about the available datasets and reviews.

[8] http://grouplens.org/datasets/movielens/1m/

[9] http://www.macle.nl/tud/LT/

[10] http://ir.ii.uam.es/hetrec2011/datasets/lastfm/readme.txt

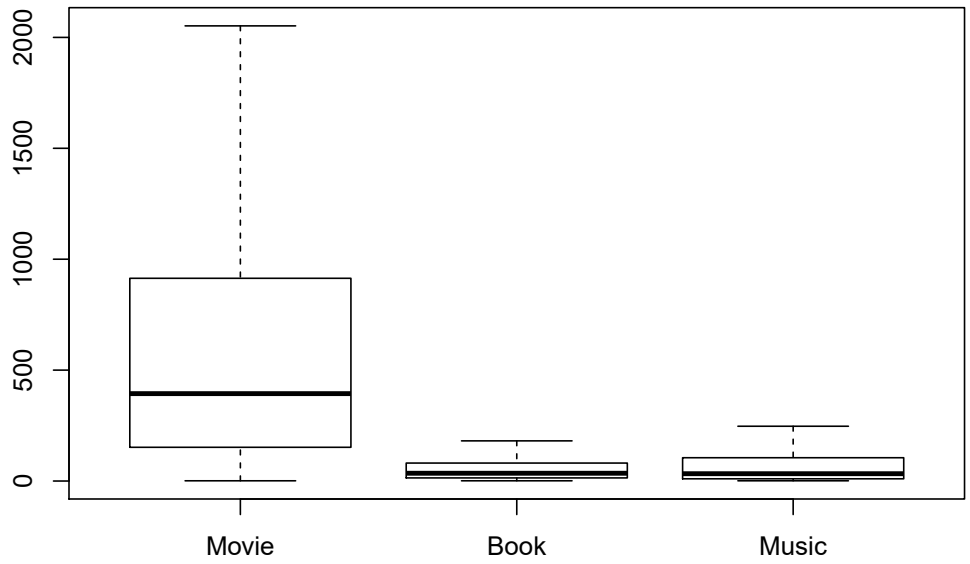


Figure 8.2: Distribution of entities extracted from the reviews per domain.

Table 8.2 lists several statistics regarding the exploited rating datasets and the analyzed reviews in the three domains considered. It is worth noting that the LastFM dataset contains a limited number of ratings with respect to the other datasets and, for this reason, it is the most sparse one. The LibraryThing dataset includes a considerable number of items, even if fewer reviews are available in the book domain. Regarding the outcome of the semantic annotation, the number of distinct and total entities identified in user reviews is reported. The ratio between these two values may be considered a measure of the variety of the mentioned topics. According to this measure, the reviews about movies are the most varied ones in terms of entities.

Figure 8.2 displays the boxplots representing the distributions of the number of annotated entities per each item according to the domain, excluding the outliers for graphical reasons. Given the interquartile range $IQR = Q3 - Q1$, all data points not belonging to the interval $(Q1 - 1.5 \cdot IQR; Q3 + 1.5 \cdot IQR)$ are considered outliers. It is clear that movie reviews are fairly different from the other ones. This may be related to the higher ratio between reviews and items in the movie domain.

We relied on a 5-fold cross-validation in order to perform the evaluations. We considered ratings positive if their score was greater than 3 on a scale from 1 to 5 for MovieLens, greater than 6 on a scale from 1 to 10 for LibraryThing, and greater than 0 for LastFM. In fact, the latter dataset contains implicit feedback, while the others are examples of explicit feedback. Exploiting the lists of the top-10 recommendations for each user, we computed the measures of precision, recall, nDCG, Entropy Based Novelty (EBN) [13], and diversity [138].

For the implementation, we rely on the LibRec library.[11] It computes measures according to the *all unrated items* protocol [121]. More specifically, it creates a top-k recommendation list for each user by predicting a score for every item not rated by that particular user, whether that item appears in the user test set or not. All the non-rated items are considered to be irrelevant for the user. This explains the low values for the measures (in particular precision and recall) as the quality of recommendations tend to be underestimated. However, Steck [121] suggests to rely on this protocol rather than the *rated test-items*, which includes only rated test items in the top-k list, as the user satisfaction regarding top-k recommendations depends on the ranking of all items.

8.3 Evaluation Results

We report the results of the first experiment on optimizing the parameters of our SemRevRec system in Section 8.3.1. The results of comparing our approach with baselines from related work are documented in Section 8.3.2.

8.3.1 Optimizing the SemRevRec Parameters

In this experiment, we evaluated the impact of the ranking function, the discovery, the occurrence threshold, and the coefficients of R3 on the performance of our algorithm. We executed SemRevRec in three domains with different ranking functions with and without the discovery phase. We also varied the configuration parameters η and κ of the ranking function R3, in order to identify possible relationships between the occurrence and the LDSD measure. Furthermore, we considered how the percentage of the minimum occurrence required for entities to be included in the candidate recommendation set impacts on the results. The main configurations tested are listed in Table 8.3.

Table 8.4, Table 8.5, and Table 8.6 summarize the results obtained with the DBpedia knowledge base in the movie, book, and music domain. For all the measures but EBN, higher values represent better results, while the lower is EBN, the higher is the novelty. The best values and configurations are highlighted in boldface.[12] For deciding if the difference between two measures was statistically significant, we relied on the Welch's t-test (or unequal variances t-test), an adaptation of the Student's t-test more reliable when the two samples have unequal variances and unequal sample sizes [114]. We considered $p < 0.001$ because we applied the Bonferroni correction as we performed pairwise comparisons.

[11] https://www.librec.net

[12] More values are highlighted for the same measure if the differences among them are not statistically significant.

Conf.	Ranking	Discovered	Occurrence	η	κ
C1	R1	False	0.05	–	–
C2	R1	True	0.05	–	–
C3	R2	False	0.05	–	–
C4	R2	True	0.05	–	–
C5	R3	False	0.05	0.50	0.50
C6	R3	True	0.05	0.50	0.50
C7	R3	True	0.05	0.75	0.25
C8	R3	True	0.05	0.25	0.75

Table 8.3: The configuration parameters of SemRevRec.

The obtained results suggest that the discovery of additional entities through Linked Data is useful for improving the precision of the recommended items. In fact, the best configurations in all the domains but music (C8 for movies, C2 for books) rely on it. In the music domain there is not a significant difference in the measures when relying on the discovery phase. This may be related to the fact that we considered reviews about musical works in order to recommend musical artists.

The best ranking function depends instead on the domain. For movies, R3 outperformed the other rankers (C8), while, for book and music recommendations, R1 accounts for the best results (C2), although in the music domain the values obtained with R1 and R2 were equivalent (C4). This suggests that a simpler ranker may be more effective on sparse data, and it could be better to rely on information from reviews than on Linked Data. Additionally, the coefficients η and κ of R3 may have a high impact on the results as shown by C6, C7, and C8 in Table 8.4, even if, in the music domain, the measures do not vary. In particular, C8 improves significantly the precision and recall measures with respect to other configurations of R3 in the movie and book domains.

Conf.	Precis.	Recall	nDCG	EBN	Divers.
C1	0.0604	0.0399	0.0412	1.2804	**0.2431**
C2	0.0529	0.0327	0.0343	1.2776	0.1629
C3	0.0604	0.0399	0.0412	1.2804	**0.2431**
C4	0.0276	0.0178	0.0197	**0.7820**	0.1716
C5	**0.0683**	0.0424	0.0491	1.0047	0.1795
C6	0.0460	0.0255	0.0320	0.9354	0.1794
C7	0.0344	0.0191	0.0243	0.8248	0.1464
C8	**0.0711**	**0.0478**	**0.0524**	1.0163	0.2114

Table 8.4: Experimental results obtained with MovieLens and DBpedia.

Conf.	Precis.	Recall	nDCG	EBN	Divers.
C1	0.0396	0.0350	0.0341	0.4081	0.7701
C2	**0.0506**	**0.0497**	**0.0465**	0.2771	0.7780
C3	0.0396	0.0350	0.0341	0.4081	0.7701
C4	0.0357	0.0340	0.0353	**0.1946**	0.8919
C5	0.0462	0.0373	**0.0462**	0.2809	0.8663
C6	0.0356	0.0331	0.0366	0.2280	0.9039
C7	0.0306	0.0269	0.0317	0.2444	0.8932
C8	0.0421	0.0418	**0.0429**	0.2077	**0.9118**

Table 8.5: Experimental results obtained with LibraryThing and DBpedia.

Conf.	Precis.	Recall	nDCG	EBN	Divers.
C1	**0.0495**	**0.0504**	**0.0486**	0.7894	0.5654
C2	**0.0504**	**0.0515**	**0.0473**	0.6640	0.6021
C3	**0.0495**	**0.0504**	**0.0486**	0.7894	0.5654
C4	**0.0504**	**0.0515**	**0.0473**	0.6640	0.6022
C5	0.0363	0.0371	0.0378	0.2619	0.9238
C6	0.0360	0.0370	0.0378	**0.2422**	**0.9325**
C7	0.0361	0.0369	0.0378	**0.2425**	**0.9325**
C8	0.0360	0.0368	0.0378	**0.2411**	**0.9329**

Table 8.6: Experimental results obtained with LastFM and DBpedia.

Figure 8.3 illustrates the performance in terms of nDCG of the three ranking functions available in SemRevRec when the number of entities considered for the recommendation process varies. The occurrence represents the minimum number of times an entity needs to be annotated in the reviews of a certain item in order to be included in the candidate recommendation set. It is expressed as a percentage of the most annotated entity for an item. The plot is based on the results obtained in the movie domain with the Wikidata knowledge base, as this can be considered the most representative case. Unsurprisingly, all rankers tend to converge, as the number of entities available decreases. However, it is important to notice that the nDCG is monotonically decreasing. This fact happens in the majority of the domains with both knowledge bases and supports the hypothesis that the higher is the number of available entities, the better is the quality of the recommendations.

Figure 8.4 compares the results obtained by the best configuration of our algorithm when using DBpedia and Wikidata for each domain. Although both knowledge bases are derived from Wikipedia, the results differ. In particular, Wikidata outperformed DBpedia in the vast majority of the considered measures. A possible

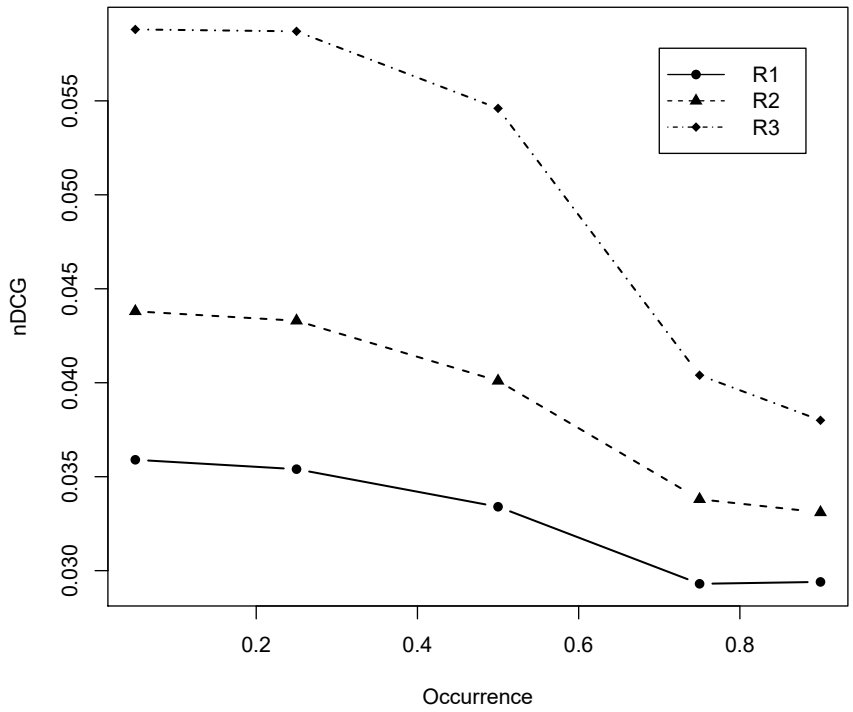


Figure 8.3: The nDCG score obtained by varying the number of entities considered with MovieLens and Wikidata.

reason may be that Wikidata provides higher data quality for the recommendation task, as it also contains knowledge manually encoded by human editors. At the instance level, this may be primary due to the interlinking of resources since we rely on the LDSD measure that exploits direct and indirect links. At the ontology level, the properties considered in the discovery may also have an high impact. We should investigate which features of a knowledge base are well suited for a Linked Data-based recommender system, although they can also depend on the particular domain considered.

Table 8.7 lists the results obtained with Wikidata. They vary significantly when the η and κ weights of the ranking function R3 are changed. Thus, we decided to include in this chapter only the results related to the configurations C4, C6, C7, and C8, although we tested all the ones listed in Table 8.3. The complete evaluation is available on the Web.[13] In general, Wikidata provides better results with respect to DBpedia and this behavior is consistent in all domains, but differences are more significant when movies are recommended.

[13] https://doi.org/10.6084/m9.figshare.5074081

Figure 8.4: A comparison between DBpedia and Wikidata. Light grey represents DBpedia, dark grey Wikidata.

Conf.	Domain	Precis.	Recall	nDCG	EBN	Divers.
C4	Movie	0.0582	0.0368	0.0438	**1.3626**	0.1223
C6	Movie	0.0757	0.0487	0.0588	1.4284	**0.1461**
C7	Movie	0.0728	0.0459	0.0552	1.4322	**0.1423**
C8	**Movie**	**0.0857**	**0.0561**	**0.0686**	1.4188	**0.1513**
C4	Book	0.0392	0.0373	0.0379	0.2634	0.8455
C6	Book	0.0452	0.0443	0.0466	0.2621	0.8705
C7	Book	0.0365	0.0334	0.0380	0.2809	0.8600
C8	**Book**	**0.0530**	**0.0530**	**0.0536**	**0.2318**	**0.8846**
C4	**Music**	**0.0536**	**0.0549**	**0.0502**	0.6319	0.6168
C6	Music	0.0384	0.0395	0.0375	**0.3083**	**0.9314**
C7	Music	0.0390	0.0401	0.0380	**0.3062**	**0.9327**
C8	Music	0.0367	0.0377	0.0363	**0.3178**	**0.9322**

Table 8.7: Experimental results obtained with Wikidata.

8.3.2 Comparison with Baselines

We compared our technique to the Most Popular, Random Guess, Item KNN, and Bayesian Personalized Ranking (BPR) [111] algorithms, as implemented in LibRec, and with SPrank [39], a state-of-the-art Linked Data-based recommender. We set the neighborhood size for Item KNN to 80, while we used 100 factors for BPR, as done by Musto et al. [95]. We configured SPrank to exploit LambdaMart as the ranking method and to follow in the DBpedia graph the same properties that we selected for our algorithm, as listed in Table 8.1.

Table 8.8, Table 8.9, and Table 8.10 list the results obtained in the movie, book, and music domain, respectively. The best values are highlighted in boldface.[14] For SemRevRec, we reported both the configuration with the best trade-off among the various measures and the best scores achieved for each measure in the experiment described in Section 8.3.1. In all the experimental trails, SemRevRec provided the best diversity and a better accuracy (both in rating prediction and ranking) than SPrank, while it improved in novelty with respect to traditional techniques. BPR accounted for the highest precision, recall, and nDCG. In general, the diversity is rather low for movies, while for music and books is above 0.6, apart for Item KNN.

[14]More values are highlighted for the same measure if the differences among them are not statistically significant. In the case of EBN and diversity, when Random Guess was the best, we also highlighted the second best because its precision, recall, and nDCG were close to zero. This means that the recommendations provided are completely unrelated and their novelty and diversity are not relevant.

The differences between SemRevRec and the other approaches are statistically significant according to the Welch's t-test with $p < 0.001$, except for SPrank, BRP, Most Popular, and Random Guess in the movie domain regarding the measure of diversity, SPrank in the book domain regarding the measures of precision and diversity, and Most Popular in the music domain regarding the measure of diversity.

Algorithm	Precis.	Recall	nDCG	EBN	Divers.
SemRevRec	0.0857	0.0561	0.0686	1.4188	0.1513
– Best Scores	0.0857	0.0561	0.0686	**0.7820**	**0.2431**
SPrank	0.0445	0.0254	0.0280	0.8813	0.1612
Item KNN	0.1626	0.1105	0.1302	2.6846	0.0696
BPR	**0.2347**	**0.1737**	**0.1930**	1.8358	0.1769
Popular	0.1325	0.0840	0.0969	2.7439	0.1412
Random	0.0055	0.0028	0.0031	**0.3018**	0.1679

Table 8.8: Experimental comparison using the MovieLens dataset.

Algorithm	Precis.	Recall	nDCG	EBN	Divers.
SemRevRec	0.0530	0.0530	0.0536	0.2318	0.8846
– Best Scores	0.0530	0.0530	0.0536	0.1946	**0.9118**
SPrank	0.0379	0.0346	0.0337	**0.1562**	0.8037
Item KNN	0.0620	0.0564	0.0662	1.4956	0.2259
BPR	**0.0862**	**0.0817**	**0.0895**	0.6043	0.7177
Popular	0.0423	0.0343	0.0447	1.6034	0.6483
Random	0.0004	0.0002	0.0003	**0.0382**	**0.9879**

Table 8.9: Experimental comparison using the LibraryThing dataset.

8.4 Discussion

In general, the results obtained by our algorithm in the music and book domains are not as good as the ones achieved with movie recommendations. This may be due to the characteristics of the reviews, as illustrated in Figure 8.2 and previously discussed. The entities annotated for each item in these two domains are much less than the entities available in movie reviews. This fact should be further studied. Furthermore, it would be interesting to investigate the impact of the number of reviews available and their quality with respect to the recommendation process. For example, a meaningful album review mentions the author and similar albums

Algorithm	Precis.	Recall	nDCG	EBN	Divers.
SemRevRec	0.0536	0.0549	0.0502	0.6319	0.6168
– Best Scores	0.0536	0.0549	0.0502	0.2411	**0.9329**
SPrank	0.0156	0.0158	0.0176	**0.1834**	0.9077
Item KNN	0.1392	0.1428	0.1720	1.6023	0.4730
BPR	**0.1545**	**0.1583**	**0.1808**	0.9404	0.6547
Popular	0.0686	0.0703	0.0791	2.0360	0.6519
Random	0.0005	0.0005	0.0004	**0.0442**	**0.9946**

Table 8.10: Experimental comparison using the LastFM dataset.

or artists the user liked, while a review describing the package is not very useful in our scenario. In fact, we aim to suggest other artists to listen to, although packaging may impact on the decision of buying a physical copy of that album. Finally, the significant difference in the results obtained when exploiting Wikidata or DBpedia suggests that the impact of knowledge bases, notably the selection of types and properties exploited, on the performance should be further analyzed.

In this work, we relied on all the reviews available for the items present in the rating datasets used for the evaluation. However, only reviews about some items, for example the ones with the average rating higher than a threshold, or only some reviews for each item, for example only the ones that are rated positively, could be considered during the semantic annotation phase. Nevertheless, lower performance on music artists and books was expected because the available ratings were more sparse than the ones regarding movies. This holds for all the algorithms and explains the general difference of scores in these domains.

SemRevRec showed the best diversity. In the sparse dataset of books, it achieved precision, recall, and nDCG comparable to Item KNN with a much higher diversity, although the former is a content-based method. Collaborative filtering techniques are known to suffer less of the overspecilization problem and provide better rating prediction and ranking than content-based ones. For this reason, although collaborative filtering is very popular, we decided to include in the baseline a technique among many, that is BPR, one of the newest and most promising. Nevertheless, it showed a lower diversity than our algorithm. Not surprisingly, it also accounted for the best rating prediction and ranking.

Our approach also provided a higher novelty than traditional techniques and a better rating prediction and ranking than SPrank. In the movie domain, SemRevRec accounted for the best novelty, while with music and books for the second best, with results close to SPrank. Additionally, when optimized for this measure, SemRevRec had similar or higher rating prediction and ranking than SPrank. On the contrary, when the former is optimized for rating prediction and ranking, it

could be preferred to the latter to increase the novelty of recommendations, while also limiting the loss in rating prediction and ranking.

Finally, SemRevRec was evaluated considering the recommendations generated for all the previous items a user liked, as its generation approach is rather naive and it takes into account only an initial item. Combining it with a machine learning technique could significantly improve its performance, but further experiments are required to prove this.

8.5 Conclusion

In this chapter we proposed a novel recommendation approach based on the semantic annotation of user reviews and Linked Data. We conducted an offline study of the recommender system in the movie, book, and music domains, which showed that our method provides the best diversity. It also improved rating prediction and ranking compared to another algorithm based on Linked Data, while it increased the novelty of recommendations with respect to traditional techniques. Furthermore, we tested our approach with different knowledge bases and Wikidata systematically achieved better results than DBpedia. Although the reviews available for the book and music domains seem to contain a smaller amount of useful information, the results of the offline study suggest that our algorithm can provide more diverse recommendations and reach an interesting compromise between the accuracy and the novelty of the suggested items.

This work represents a practical application of multicriteria evaluation approaches, but it also raises further interesting research issues that still need to be properly addressed. For this reason, we intend to investigate in greater details how the nature of the user reviews influences the performance of our algorithm. Furthermore, the significant difference in the results obtained when exploiting Wikidata or DBpedia suggests that too little is known about how knowledge bases (notably their types and properties) might impact on the performance of Linked Data-based recommender systems. We also plan to take into account the sentiment of the reviews, that is whether the overall opinion on the item reviewed is positive or negative. Finally, we are evaluating applications of our approach on textual resources different than reviews, for example research papers or their abstracts. In this case sentiment would not be relevant, while annotated entities could be concepts representing the main topics addressed in the document.

Chapter 9

Second Use Case: Music Recommender System

In recent years, music streaming services strongly modified the way in which people access to music content. In particular, the music experience does not foresee anymore to follow pre-defined collections of tracks edited by music artists or labels: the end-user is now free to produce her own playlist with potentially unlimited freedom. As a consequence, the automatic playlist generation and continuation are now crucial tasks in the recommender system field.

This chapter describes the results obtained by the D2KLab team, lead by the author of this dissertation, for the task of playlist completion obtained in the context of the RecSys Challenge 2018. This work relies on an ensemble strategy which involves different types of features, including sequential embeddings, title embeddings and lyrics features. Therefore, the proposed approach could be considered a multicriteria sequence-based recommender system. Following the challenge rules,[1] the target dataset is the Million Playlist Dataset (MPD), which contains metadata for 1 million playlists gathering more than 2.2 million distinct tracks. The implementation of our approach is publicly available on GitHub.[2]

The remainder of this chapter is structured as follows: Section 9.1 presents our ensemble approach, while Section 9.2 details the design of the Recurrent Neural Networks. In Section 9.3 we discuss the intuition behind the implementation of Title2Rec. Section 9.4 explains the optimization conducted on the ensemble, the RNN, and Title2Rec. We describe the experimental results in Section 9.5. Finally, in Section 9.6, we provide the conclusions.

[1] https://recsys-challenge.spotify.com/rules

[2] https://github.com/D2KLab/recsys18_challenge

9.1 Ensemble

Our approach builds upon an ensemble voting strategy of different runs of multiple Recurrent Neural Networks (RNNs) and one execution of Title2Rec. The RNNs are configured differently in terms of network inputs and hyper-parameters. The RNNs are used to predict the missing tracks to be part of a playlist and thus assume to have seed(s) track(s) of the playlist to be utilized as initial elements of the network bootstrap (Section 9.2). However, when only the title of the playlist is available, our approach relies on a fall-back strategy that implements a K-means clustering of the playlists and a word embedding model of their titles (trained with fastText), called Title2Rec (Section 9.3). Figure 9.1 illustrates the overall approach.

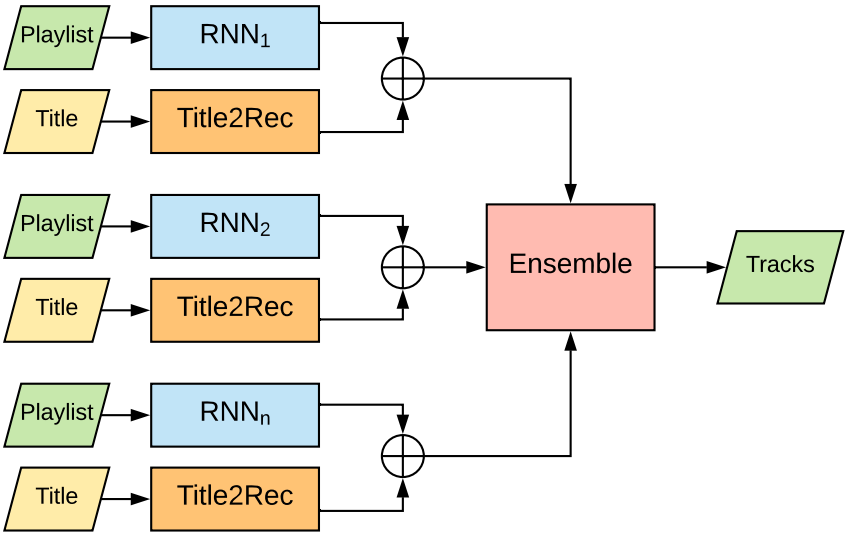


Figure 9.1: The proposed ensemble architecture for playlist completion. The inputs are a playlist and its title.

The ensemble weighs the rankings of the different runs by giving more importance (more weights) to the top ranked tracks and less to the low ranked tracks, similarly to a Borda count election. In detail, given a ranked set of predictions coming from a configuration k, corresponding to a particular configuration of the RNN jointly combined with Title2Rec, $R_k = \{T_1, T_2, \ldots, T_{500}\}$, we assign to each track a score s_k that has its maximum for the first track in the ranking and minimum for the last one, i.e. $s_k(T_i) = 500 - i + 1$. Then, we sum the scores over all the configurations that we want to ensemble, obtaining a final score for each track $s(T_i) = \sum_k s_k(T_i)$ that we use to create the final ranking of the tracks. Take as an example (with 3 tracks instead of 500 in the predictions) a configuration 1 with ranking $R_1 = \{T_1, T_2, T_3\}$ and a configuration 2 with ranking $R_2 = \{T_1, T_3, T_2\}$. We would get $s_1(T_1) = 3$, $s_1(T_2) = 2$, $s_1(T_3) = 1$, $s_2(T_1) = 3$, $s_2(T_3) = 2$, $s_2(T_2) = 1$ and thus $s(T_1) = 3 + 3 = 6$, $s(T_2) = 2 + 1 = 3$, $s(T_3) = 1 + 2 = 3$, obtaining as a

final ranking $R = \{T_1, T_2, T_3\}$, or equivalently $R = \{T_1, T_3, T_2\}$ as T_2 and T_3 have the same score.

9.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are one of the most commonly used typology of neural networks [78]. In recent years, thanks to advancements in their architecture [64, 30] and in computational power, they have become the standard to effectively model sequential data. They have been used successfully for tasks such as sentiment analysis [126], speech recognition [54], image captioning [74], predicting tourist paths [100] and neural language models [87]. One of the typical applications of RNNs is language modeling, i.e. the task of learning a probabilistic model of text in order to generate new text by recursively predicting the next word in a sentence [124]. We use RNNs, more specifically Long-Short Term Memory (LSTM) cells [64], in a similar vein to the language modeling problem, i.e. training the network to predict the next track in a playlist and sampling tracks from the learned probability model to generate predictions. In practice, rather than using only the track as input, we use a richer representation that also exploits the artist, the album, the title and, possibly, lyrics features (Figure 9.2).

In the following sections, we describe in detail the input features as well as the generation strategy.

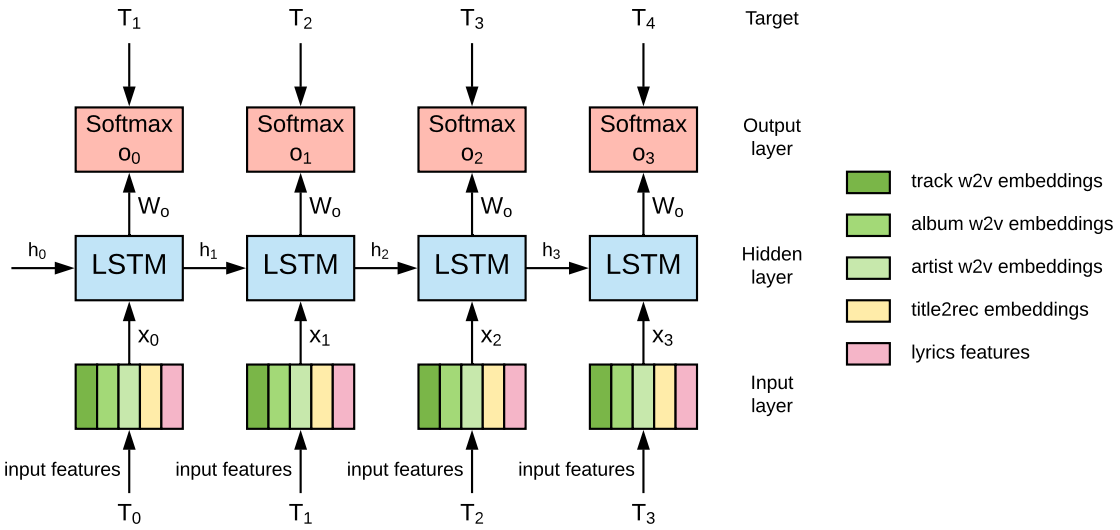


Figure 9.2: Our RNN architecture for playlist completion. The input vectors include word2vec embeddings for the track, the album, and the artist, a fastText embedding for the playlist title and numerous features extracted from the lyrics.

9.2.1 Input Vectors

Track, Album and Artist Embeddings

In order to leverage the information in the dataset concerning tracks, artists and albums, we opt for an approach based on word2vec [86] embeddings. More precisely, we train the word2vec model separately on sequences of tracks, albums and artists in the order of appearance in the playlist, obtaining three separated word2vec models encoding co-occurrence patterns of tracks, albums and artists respectively. Each word2vec model is based on the Skip-gram model with negative sampling using default hyper-parameters of the Gensim implementation [109]: embedding vector dimension is $d = 100$, learning rate $\alpha = 0.025$ linearly decaying up to $min_\alpha = 0.0001$, window size $c = 5$, number of epochs is $\eta = 5$.

We concatenate the three representations of the tracks, albums and artists, obtaining an input vector x_{w2v} whose dimensionality is $|x_{w2v}| = 300$.

Title Embeddings

The title of a playlist can potentially contain interesting information about the intention and the purpose of its creator. The title can suggest that the tracks in certain playlist are intended to suit a certain goal (e.g. *party*, *workout*), a mood (*sad songs*, *relaxing*), a genre (*country*, *reggae*), or a topic (*90's*, *Christmas*). Our intuition, supported by the experiments described later in this section, is that playlists with similar titles may contain similar tracks. The title similarity could rely on pre-trained models and thesauri. However, we opted for computing a model that is specific for the playlist continuation task, using the sole data of the MPD.

A playlist embedding p_{w2v} is computed as the mean of the embeddings of the tracks composing the playlist, as generated in the previous section. The playlist embeddings are then grouped in n clusters, applying the K-means algorithm [58].

We empirically observed that, apart from very general clusters, we also created clusters containing specialized playlists, obtaining as a consequence groups of titles that belong to the same semantic area. For example, a cluster contains playlists like *Christmas feels*, *December* or with titles including the emoji of Santa Claus, while another group encompasses playlists like *country* and *Alabama*.

Each cluster c expresses a composed label, which is the concatenation of the titles of all the playlist $p \in c$ separated by a blank space. These labels can be seen as a corpus of n documents (one for each cluster) that is used as input for the fastText algorithm [72]. Because this algorithm is able to represent textual information at the level of n-grams from 3 to 6 character, the Title2Rec model in output computes the embeddings of any playlist title, being this already seen in the dataset or totally unknown. Figure 9.3 illustrates the process of the Title2Rec model generation.

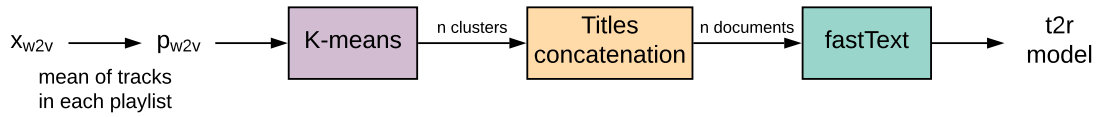


Figure 9.3: The pipeline for generating the title embedding model used in Title2Rec. The embeddings are computed through a fastText model trained on a corpus of concatenated titles of similar playlists.

Lyrics Embeddings

Since playlists contain tracks that share semantic properties (such as the genre) and acoustic properties (such as the mood), we hypothesize their lyrics share features as well. To this end, we extract numerous features from the lyrics for a large set of tracks used in the MPD dataset ($v \in \mathbb{R}^n$) that describe different stylistic and linguistic dimensions of a song text:

- *vocabulary* ($v \in \mathbb{R}$): as a measure of the vocabulary richness, we compute the type-token ratio of a song text.
- *style* ($v \in \mathbb{R}^{27}$): to estimate the linguistic style of a song text, we measure the line lengths (in characters and in tokens) and the frequencies of all major part-of-speech tags. We further count rhyme occurrences and "echoisms" (sung words like "laaalala" and "yeeeeeeeaaaaaaah").
- *semantics* ($v \in \mathbb{R}^{60}$): we build a topic model with 60 topics on the song text bag of words using Latent Dirichlet Allocation [17]. Each song text is then represented by its association to these topics.
- *orientation* ($v \in \mathbb{R}^3$): this dimension models how the song narrative (entities, events) is oriented with respect to the world. We encode a temporal dimension, i.e. whether the song mainly recounts past experiences or present/future ones, by representing the fraction of past tense verb forms to all verb forms as a feature.
- *emotion* ($v \in \mathbb{R}^6$): we model the subjectivity (subjective vs. objective) as well as the polarity (positive vs. negative) of the song text. Furthermore, the emotions conveyed are modelled in a common two-dimensional model that accounts for degrees of arousal and valence.
- *song structure* ($v \in \mathbb{R}^4$): as a proxy of the structure of the lyrics, we use the line lengths as well as the lengths of paragraphs in the song text.

For experimental purposes, we grouped the previous features in two categories:

- *deterministic* ($v \in \mathbb{R}^{23}$): it encompasses all features generated in a deterministic way such as features related to the structure, the vocabulary, and the style of the lyrics. We excluded from this group the frequencies of part-of-speech tags, as they depend on the tagger used.
- *fuzzy* ($v \in \mathbb{R}^{18}$): it includes the features generated in a non-deterministic fashion such as orientation, emotion, and the frequencies of POS tags.

All features are scaled using a custom feature scaler that combines two elements. It accounts for outliers by scaling the data non-linearly based on the percentile of the feature value distribution they belong to. Finally, it scales the data linearly to the same $[-1,1]$ interval that non-lyrics features live in.

Retrieving lyrics for the MPD dataset is achieved by linking it to the WASABI corpus [85].[3] The WASABI corpus is an ongoing resource that contains 2.1M song texts (of 77k artists), and for each song it provides the following information: the lyrics extracted from `http://lyrics.wikia.com`, the synchronized lyrics (when available) from `http://usdb.animux.de`, DBpedia abstracts and categories the song belongs to, genre, label, writer, release date, awards, producers, artist and/or band members, the stereo audio track from Deezer (when available), the unmixed audio tracks of the song, its ISRC, BPM, and duration. In total, we linked 416k tracks in MPD (out of 2.2M unique tracks) to WASABI tracks that contain the lyrics. While the linked tracks proportion with ∼20% seems small, the linked tracks cover 53% of all 66M track occurrences in MPD because of the typical fat-tailed distribution, where some songs are extremely common while most titles occur only rarely in a playlist. Linking the lyrics was done in three levels of accuracy: direct Spotify URI matching gave us 155k links, exact artist and title matching provided 334k matches, and finally lower casing and deleting bracketed content (in song titles only) led to 51k matches. As the results overlap we ended up with 416k matched tracks in total. Some of our lyrics features are language-specific, so we decided to compute lyrics features exclusively on English song texts. This finally resulted in 367k English song texts we computed lyrical features on. Language detection is done with the *langdetect* package[4] and datasets of MPD and WASABI are merged along the axes of their Spotify URIs, artist names, song title names, respectively.

9.2.2 Learning Model

As mentioned earlier, we address the problem of playlist continuation as a language modeling problem. More specifically, we train the RNN to predict the next track in a playlist, defining the targets Y to be the inputs X shifted in time, i.e.

[3] `https://wasabi.i3s.unice.fr`

[4] `https://github.com/Mimino666/langdetect`

$X = \{(\hat{T^j}_0, \hat{T^j}_1, \ldots, \hat{T^j}_{N_j-1})\}$ and $Y = \{({T^j}_1, {T^j}_2, \ldots, T^j_{N_j})\}$ where $\hat{T}$ represents a track and its metadata (artist, album, playlist title, lyrics features), T represents a track id in a playlist, $j = 1, \ldots, M$ is a playlist index and N_j is the length of the j-*th* playlist. In this way, we train the model to learn a probability distribution of the next track $P(T_N|\hat{T}_{N-1}, \hat{T}_{N-2}, \ldots, \hat{T}_0)$ given the previous ones, which is parametrized by the network outputs that are converted into probabilities by the final softmax layer (Figure 9.2). The training algorithm attempts to minimize the cross-entropy loss function L, that measures the disagreement between the learned probability model and the observed probability model of the targets Y. The perplexity metric that is reported in the experiments is similar to the one detailed in Section 4.2.2 and it corresponds to $ppl = 2^L$. In practice, rather than using probabilities, we use the 'logits' p_i where i is a track index, un-normalized scores that are proportional to the probabilities. Different optimization algorithms to minimize the loss are empirically compared to select the most appropriate one.

9.2.3 Generating Predictions

We experiment three different strategies to generate track predictions from the RNN. Given an input seed and the hidden state, the trained model outputs the logits p_i, i.e. un-normalized scores that are proportional to the probability that a given track appears after the sequence of seeds s. In details, we considered the following approaches, as depicted in Figure 9.4.

`do_sample` It samples the track with the highest logit p_i, where $\hat{i} = arg\ max(p_i)$, given the set of seeds s. It adds the sampled track $\hat{i}$ to the seeds s, then it repeats the previous operations until 500 tracks are sampled.

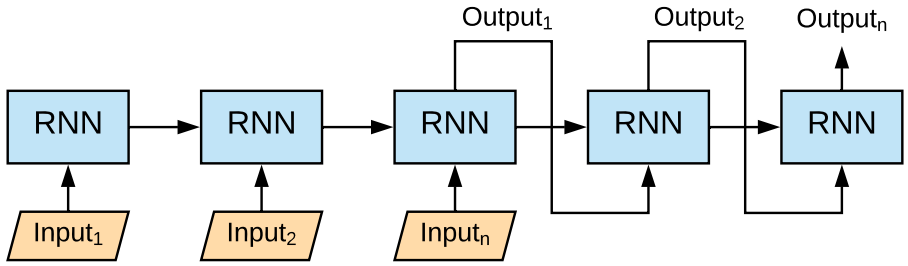
`do_rank` It ranks the tracks according to their logit value p_i, given all the seeds s, then it selects the top-500 tracks with the highest logit.

`do_summed_rank` It computes the logits p_i for every seed. It averages all the logits in the sequence obtaining $\hat{p_i}$ and then it ranks the tracks according to the values of $\hat{p_i}$.

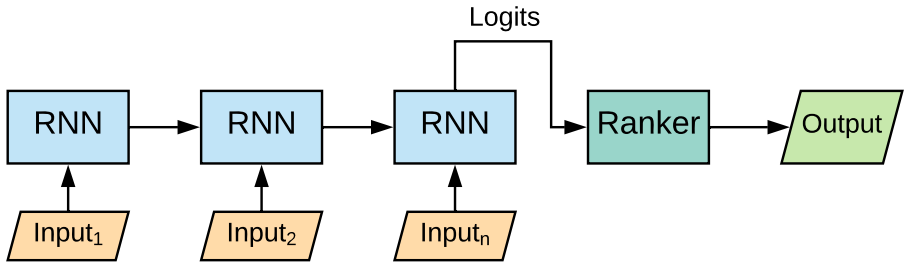
9.3 Title2Rec

Title2Rec recommends tracks taking as input the playlist title, following the procedure illustrated in Figure 9.5. The title is translated into a vector p_{t2r}, named title embedding, computed by applying the strategy described in Section 9.2.1 to the playlists defined in the MPD dataset.

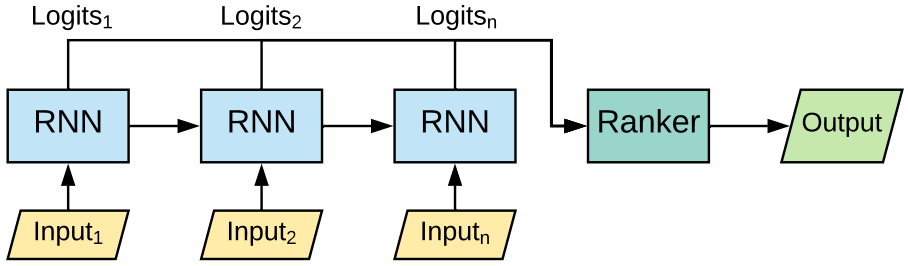
Given a new seed playlist, we compute its title embedding in the same way. Then, we select a subset P including the top-300 most similar playlists to the given



(a) `do_sample`



(b) `do_rank`



(c) `do_summed_rank`

Figure 9.4: Our three strategies for generating track predictions.

one by comparing its embeddings with p_{t2r} using the cosine similarity. Finally, the required number of tracks are selected among the ones available in P. The tracks have been ordered to ensure that the most popular ones in P are placed at the top of the list.

9.4 Optimization

In the following, we describe the empirical evaluations conducted with the purpose of optimizing the configuration of the RNN, Title2Rec, and the ensemble.

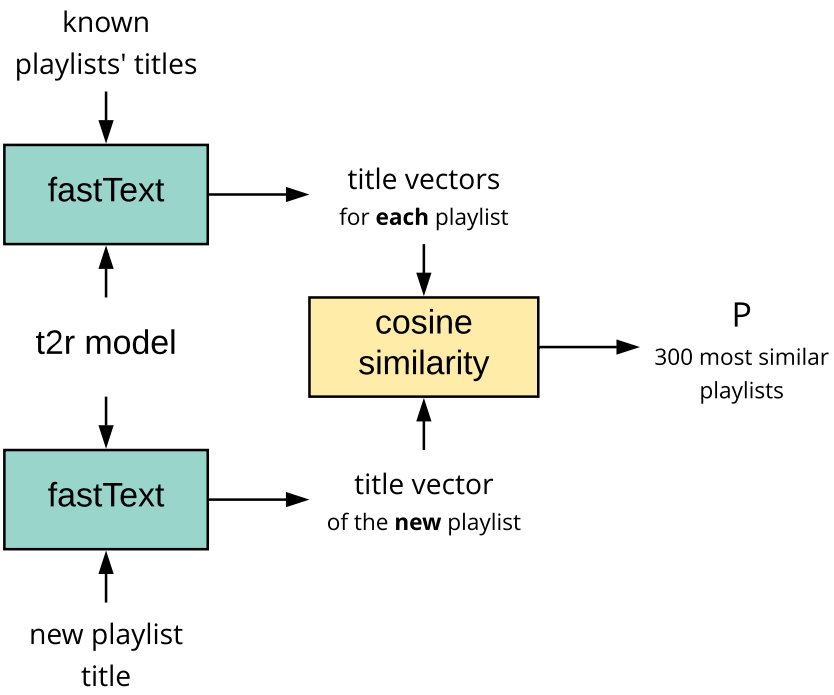


Figure 9.5: The Title2Rec algorithm compares the fastText representation of the title of a seed playlist to the known ones using the cosine similarity.

9.4.1 RNN Optimization

For optimizing the hyper-parameters of the RNN, we executed a grid search on a down-sampled version of the MPD dataset containing 100,000 playlists. We considered the following parameters:

- optimizer: $opt = \{Gradient,\ RMSProp,\ ADAM\}$
- learning rate: $lr = \{1,\ 0.5,\ 0.1,\ 0.01\}$
- number of steps: $ns = \{10,\ 20\}$
- hidden layer size: $hl = \{50,\ 100\}$

For each configuration (opt, lr, ns, hl), we trained the RNN model and we measured its perplexity on a validation set consisting of 1,000 playlists. Furthermore, we measured its R-Precision, nDCG, and Click metrics as defined in the challenge rules on a separate test set of the same size. The validation and test sets used for optimization purposes contain playlists with the first 5 tracks available as the initial seed, while the others are hidden.

We considered a total of 48 possible configurations: the values of perplexity of the most significant ones are reported in Table 9.1. Perplexity measures the 'surprise' of the probabilistic model in observing the data and it is defined as s^L where L is the cross-entropy loss function. Thus, lower values of perplexity corresponds to better models. We observe that, when the hidden size is fixed, the best performing optimizer is ADAM. Furthermore, increasing the number of steps reduces the perplexity of the RNN, but it does not have a significant effect on the R-Prec.

Finally, because of time constrains, we selected the configuration (*ADAM*, 1, 10, 50) as the optimal one, despite its higher perplexity: in fact, we empirically observed that a smaller hidden size results in a shorter training duration.

Optimizer	L.R.	Steps	Hidden	ppl	Time	R-Prec.
ADAM	1	20	100	1357.04	3:29	0.1739
ADAM	1	10	100	1482.86	3:39	0.1742
Gradient	1	10	100	1693.96	3:32	0.1566
ADAM	1	10	50	1716.92	2:30	0.1745
Gradient	1	10	50	2005.54	2:25	0.1543

Table 9.1: The results of the most significant RNN models. 'L.R.' stands for learning rate, 'Steps' for the number of time steps, 'Hidden' for the size of the hidden layer, 'ppl' stands for perplexity, 'Time' is the training time in hours:minutes.

We evaluated in a controlled setting all the strategies for generating the recommended tracks described in Section 9.2.3. We observed that, independently from other hyper-parameters, the technique called *do_summed_rank* systematically achieved better results than the other ones in all the metrics considered. For this reason, we selected this algorithm as our track generation strategy.

Finally, we analyzed the effects on the evaluation metrics of the different categories of features extracted from the lyrics as defined in Section 9.2.1, and we selected the groups emotion and fuzzy as the most performing ones.

9.4.2 Title2Rec Optimization

In order to improve the performances of Title2Rec, we worked on different parts of the pipeline. Each optimization has been tested by running the algorithm on a validation set of 1,000 playlists. Then, only the edits that improved the scores with respect to the non-optimized version have been kept in the final version.

We applied a pre-processing on each single title that performed a series of tasks:

- lowercasing;
- detecting and separating emoji from words;

- separating the skin code from the emoji;
- detecting and separating emoticons from words;
- transforming space-separated single letters into words (e.g. "*w o r k o u t*" becomes "*workout*";
- remove '#' from hashtags.

Other tasks that have been tested with no improvements are:

- detecting and separating punctuation from words;
- removing stop words;
- removing all spaces.

The latter point has been partially exploited because we noticed an improvement in the results by including in the corpus both versions of the title, that is keeping the spaces (as in "green day") and removing them ("greenday").

Another optimization step included the usage of different parameters for executing the pipeline. The clustering phase have been tested with different values of k (the number of clusters in output for the K-means algorithm). The value of 500 gives better results than smaller and bigger ones, which produce clusters that are respectively less specialized and less populated. The fastText training has been run with 5 epochs, a learning rate of 0.1 and different loss functions (ns, hs, *softmax*), window sizes (3, *5*, 10). The values in italics represent the best results.

The ordering by popularity described in Section 9.3 has been modified so that the impact of each playlist is proportional to the similarity of its title to the seed. In other words, a track has a higher chance to be recommended if it is included in a large number of playlists in P and if most of them are among the top ones more similar to the seed.

Finally, some improvements come from the inclusion of the playlist descriptions in the training. On the whole set of descriptions in the MPD dataset, we compute a TF-IDF model. Thanks to this, we are able to extract a set of keywords for each description by selecting the three words with the highest score. These keywords are added to the documents used to build the clusters. The contribution of the description is null when the playlist does not include any.

9.4.3 Ensemble Optimization

We studied the performance of the ensemble by applying a combination without repetition sampling of different runs for each of the tracks, namely main and creative, and for different groups of runs. In detail, given n the total number of

runs, and k the grouping factor, we devised a number of $\frac{n!}{k!(n-k!)}$, where we varied $k = 1, \ldots, n-1$. We then selected the best performing configuration for both the main and the creative tracks by optimizing the three metrics used for the final ranking. These configurations are reported in Section 9.5.

9.5 Experimental Results

In order to evaluate the effectiveness of our approach, we have divided the official MPD dataset in a training, a validation, and a test set. The validation and the test set contain 10,000 playlists each, that is the 1% of the original dataset. These playlists have been selected according to the characteristics of the MPD provided by Spotify.[5] Thus, the validation and test playlists are divided into 10 different categories: each of them defines a peculiar way of hiding some information during the testing phase, i.e. the number of seed tracks or their order.

Furthermore, we have implemented an evaluation tool that computes on our split the same metrics that are described in the challenge rules. Following this approach, it is possible to inspect the evaluation results for each category of the test set separately. As expected, the category containing playlists with only their title and no tracks proved to be the most difficult one to address.

Table 9.2 contains the results obtained on our test set by Title2Rec, Word2Rec, and the RNNs trained with different optimizers and input vectors. Word2Rec corresponds to the word2vec model trained on sequences of tracks as described in Section 9.2.1 and used to generate predictions directly by looking up the 500 most similar tracks to the seeds. All the neural models, but the first two, were trained with the optimal configuration described in Section 9.4.1. These models are computationally demanding: the training phase lasted more than three days per epoch. The numbers 300 and 400 represent the dimensionality of the input vectors: the 300 models were trained without the title embeddings, while the 400 ones also exploit the fastText model described in Section 9.2.1. All the RNNs that include the features extracted from the lyrics were trained with input vectors of dimensionality higher than 400.

Table 9.3 lists the results computed on our test set for the best performing configurations in the two tracks of the challenge. The models combined in the ensemble are the following:

Main track RNN 300 (Gradient; Epoch 1 and 2), RNN 300 (ADAM; Epoch 1 and 2), and RNN 400 (Epoch 1 and 2).

Creative track RNN 300 (Gradient; Epoch 1 and 2), RNN 300 (ADAM; Epoch 1

[5] https://recsys-challenge.spotify.com/challenge_readme

Approach	Optimizer	Epoch	R-Prec.	nDCG	Click
Title2Rec	-	-	0.0837	0.1260	12.007
Word2Rec	-	-	0.0963	0.1444	8.4322
RNN 300	Gradient	1	0.1417	0.1621	4.1902
RNN 300	Gradient	2	0.1500	0.1656	3.9433
RNN 300	ADAM	1	0.1557	0.1702	3.9213
RNN 300	ADAM	2	0.1457	0.1672	4.4224
RNN 400	ADAM	1	0.1572	0.1708	3.9340
RNN 400	ADAM	2	0.1520	0.1694	4.1307
RNN Emotion	ADAM	1	0.1556	0.1702	4.0101
RNN Emotion	ADAM	2	0.1500	0.1680	4.3594
RNN Fuzzy	ADAM	1	0.1555	0.1698	3.9950
RNN Fuzzy	ADAM	2	0.1503	0.1683	4.3456

Table 9.2: Experimental results of different approaches on our test set.

only), RNN 400 (Epoch 1 and 2), RNN Emotion (Epoch 1 and 2), and RNN Fuzzy (Epoch 1 and 2).

Track	R-Precision	nDCG	Click
Main	0.1611	0.1710	3.6349
Creative	0.1634	0.1717	3.5964

Table 9.3: Experimental results of the ensemble on our test set.

9.6 Conclusion

Completing automatically playlists with tracks contained in the MPD dataset is a particularly difficult task due to the dataset dimension and the variety of playlists generated by numerous users having different likes and behaviors bringing great diversity. In this chapter, we presented the D2KLab recommender system that implements an ensemble approach of multiple learning models differently optimized combined with a Borda count strategy. Each model runs an RNN that exploits a wide range of playlist features such as artist, album, track, lyrics (used for the creative track), title and a so-called Title2Rec that takes as input the title and that is used, as fall-back strategy, when playlists do not contain any track. The approach showed to be robust in such a complex setting demonstrating the effectiveness of learning models for automatic playlist completion.

The experimental analysis brought to further attention three points, namely the generation strategy, complementarity of the learning models, and computing time. The generation strategy has a great impact on the results and it pointed out that a recurrent decoding stage is less performing than using a ranking strategy that weighs the output of each RNN of the encoding stage. The ensemble strategy aggregates different outputs of the learning model runs by pivoting the generated ranking. This has granted a sensible increment in performance, so we plan to study further the complementarity of the runs and to build a learning model to automatically select the best candidates. Finally, the computing time has been a crucial experimental setup element due to the generation of the RNN learning model; we addressed it by creating different sizes of the MPD dataset randomly selected and by optimizing the learning models on the hardware at disposal, becoming another factor of differentiation for shaping a performing submission.

Chapter 10

Conclusion and Future Work

This dissertation explored different challenges related to the offline evaluation of sequence-based and top-k recommender systems. We proposed to adopt a multicriteria approach to mitigate the popularity bias introduced by many rating datasets and a robust evaluation protocol to ensure the reproducibility of the results. In particular, we considered three main research lines: identifying the most appropriate metrics to evaluate a sequence-based recommender system, creating a protocol suitable for comparing ranked lists of suggestions, and analyzing the structure of rating datasets to understand their impact on the results of an offline trail.

We proposed two evaluation protocols by formalizing their theoretical backgrounds and by also developing their software implementations. While relying on them, we studied a possible technique to visualize the internal structure of any rating dataset and we designed a method for generating synthetic collections of user preferences that could be successfully exploited to conduct offline evaluations. Finally, we applied our knowledge of multicriteria approaches to different use cases, by designing novel recommendation algorithms and assessing their performance.

In detail, the main contributions of this dissertation are the following:

- A systematic literature review about multicriteria recommender systems, that was reported in Chapter 3.
- Sequeval, an offline evaluation protocol for sequence-based recommenders, introduced in Chapter 4.
- A distributed approach to assess the quality of top-k lists of suggestions, called RecLab and discussed in Chapter 5.
- RS-viz, a method for visualizing with a 3D scatter plot the ratings available in a dataset, that was presented in Chapter 6.
- A clustering technique capable of generating synthetic datasets in a realistic way starting from already existing ones, as shown in Chapter 7.

- The evaluation of a recommender based on the entities mentioned in the reviews of the suggested items, described in Chapter 8.
- The creation of an RNN-based algorithm to suggest sequences of songs to be added to a playlist, that was outlined in Chapter 9.

The systematic literature review explored the topic of multicriteria recommenders, as defined by the structure of their users' preferences. We investigated the approaches currently available in literature and how they were evaluated, considering the protocols, the metrics, and the datasets. We discovered that it is not possible to directly compare different algorithms due to the variability in the evaluation methods adopted by the reviewed studies. Furthermore, the lack of publicly available rating datasets emerged as another critical point.

Sequeval is an experimental protocol for performing the offline evaluation of sequence-based recommender systems. It exploits a multicriteria approach to analyze the suggested sequences from different angles, considering eight metrics. Therefore, the experimenter can select the most promising techniques according to the dimensions she considers the most relevant in a domain of interest. We conducted different experimental campaigns by relying on this framework and we observed that the results are adequate to identify the straights and weaknesses of the recommendation algorithms under investigation.

A similar multifaceted set of metrics was exploited for creating RecLab, a method to evaluate top-k recommender systems in a distributed fashion. By relying on widespread Web protocols, it is possible to assess in a reliable way different algorithms without exposing their implementation details. Please note that both frameworks are not bound to any particular rating dataset or splitting strategy. We empirically observed the effect of the configuration parameters of RecLab on the experimental results considering different domains and recommenders.

To better interpret the results of an evaluation campaign, we proposed and prototyped RS-viz, an interactive visualization method for understanding the structure of the preferences available in a dataset. From the resulting plots, it is possible to intuitively observe unexpected statistical distributions, and, thus, being aware of the probable presence of an associated bias in the suggested items. We validated this hypothesis considering different versions of the LastFM dataset and the numerical results obtained from them with multiple recommenders.

Because of the scarcity of publicly available collections of ratings highlighted by our systematic literature review, we designed a method to generate synthetic datasets that exhibit the same properties of existing ones. By relying on RecLab, we verified that this approach is useful to anonymize potentially private ratings while preserving the possibility of using them to successfully train a recommender system. In particular, we observed that the experimental results obtained when relying on the generated datasets are consistent with the ones of the reference datasets.

We applied a multicriteria evaluation protocol to a recommender system based on the semantic annotation of user reviews, named SemRevRec. The motivating idea of this content-based approach is that users could explicitly mention items that are unexpected but also related with the initial one. Our offline study conducted in multiple domains showed that this method provides the best diversity among the considered approaches, while also increasing the precision of the suggestions with respect to another method based on Linked Data.

Finally, in the context of the RecSys Challenge 2018, we designed a novel sequence-based recommender system based on Recurrent Neural Networks and text embeddings. The goal of this approach is to suggest how to complete a music playlist starting from a few seed songs and the title of the playlist. The proposed method was evaluated in a multicriteria fashion considering three metrics that were defined by the organizers of the challenge. Despite the high competitiveness of this field, our approach has been ranked in the first third of the leaderboard.

The complete list of the publications describing the studies discussed in this dissertation is available in Appendix B.

10.1 Limitations

In the systematic literature review reported in Chapter 3, we considered as multicriteria only the recommender systems that are capable of exploiting a dataset containing multiple ratings for each user–item pair, thus representing more complex user preferences. This strict choice was necessary to clearly define the scope of our investigation. Nevertheless, in the remainder of this dissertation we focused our attention on the experimental approaches for assessing existing algorithms and we mainly considered "multicriteria" as an evaluation technique based on multiple measures. Furthermore, the recommender discussed in Chapter 9 could be considered a multicriteria one with respect to the content-based attributes of the items.

Both Sequeval and RecLab rely on several recommendation performance objectives. However, the availability of many metrics may produce results which are difficult to interpret, especially if we are uncertain of what are the most relevant dimensions in our recommendation scenario. For this reason, it would be useful to define a way for summarizing the outcome of an evaluation campaign. More in general, this is a common limitation of offline experiments, and it needs to be addressed by analyzing the most promising algorithms in a subsequent online trail.

Our visualization toolkit RS-viz was empirically validated by comparing the plots obtained from alternative versions of a rating dataset with the corresponding results of two offline trails involving multiple algorithms. However, we should confirm these results by conducting a user study to investigate if researchers and practitioners are able to correctly use it to explain the performance of different recommender systems in a particular domain.

The approach proposed to generate synthetic datasets starting from an existing one is based on the K-means clustering algorithm. Even if we studied the impact of the value of K on the recommendation results, additional work is required to better understand what is the optimal value for a certain domain. Furthermore, we should also investigate what is the effect of artificially increasing the number of users during the generation phase.

10.2 Future work

The offline comparison of different recommender systems is a challenging task that can be successfully completed only by understating the structure of the rating dataset, the impact of the evaluation protocol, and the meaning of the exploited metrics. Even if in an industrial setting the ultimate method for assessing a recommendation algorithm is studying its impact on company profits, there is anyway the need of conducting offline experiments for comparing a large number of alternatives or selecting the most promising configuration parameters.

This dissertation discussed an evaluation framework for sequence-based recommender systems, a distributed approach for analyzing top-k ranked lists of suggestions, a method for visualizing datasets of user preferences, and a generative approach for creating synthetic ratings. Nevertheless, the outcomes of these studies pose further research challenges that should be address by future works.

Regarding both evaluation toolkits, we plan to study in more depth what are the relationships among the different metrics included in the frameworks, with the purpose of integrating them in a final value that expresses the overall quality of the recommender. Such a global score should be related to the recommendation scenario: for example, diversity may be important when recommending POIs to a tourist, but less useful in the music domain.

Furthermore, it would be desirable to be able to create evaluation frameworks that are adopted by a community of researchers when testing their algorithms, harmonizing the evaluation protocols and the interpretation of the performance of the analyzed recommender systems. For this reason, it is necessary to identify and to include in them some additional meaningful datasets, related to different domains that could be exploited during the evaluation phase, as well as other baselines and novel recommendation methods.

Finally, we would like to expand our evaluation frameworks to also support the online experimentation that should be performed after the offline analysis. The final goal of this dissertation is, in fact, to enable researchers to spend more time in realizing the recommendation algorithm as they can rely on an evaluation protocol that has already been designed and validated.

With respect to RS-viz, we would like to quantitatively characterize rating datasets according to different dimensions and place them in various categories,

for example by analyzing the diversity of user preferences or the tendency to rate popular items only. This empirical categorization would enable the users of our tool to better understand the ratings available and to select the most appropriate recommendation approach according to such proprieties.

Furthermore, we would like to improve RS-viz by developing other visualization methods to enable more comprehensive analysis. Finally, additional studies are needed to better understand how the proposed approach could be extended for also visualizing non-conventional datasets, for example the ones enhanced with context-aware information like spatial and temporal data.

Finally, there is the need of exploring additional methods for creating synthetic datasets. We believe that Generative Adversarial Networks (GANs) could be successfully exploited for this task, as they are already used to generate fake images starting from real ones [53]. Such approaches would require the definition of a way for representing the preferences of a user similarly to an image.

Appendix A

Systematic Literature Review

Code	Author	Title	Year	Publication	Source
P1	Liu, L.; Mehandjiev, N.; Xu, D.-L.	Multi-criteria service recommendation based on user criteria preferences	2011	Fifth ACM Conference on Recommender Systems	ACM Digital Library
P2	Shambour, Q.; Lu, J.	A hybrid multi-criteria semantic-enhanced collaborative filtering approach for personalized recommendations	2011	International Conferences on Web Intelligence and Intelligent Agent Technology	ACM Digital Library
P3	Jannach, D.; Karakaya, Z.; Gedikli, F.	Accuracy improvements for multi-criteria recommender systems	2012	13th ACM Conference on Electronic Commerce	ACM Digital Library
P4	Hdioud, F.; Frikh, B.; Ouhbi, B.	Multi-criteria recommender systems based on multi-attribute decision making	2013	International Conference on Information Integration and Web-based Applications & Services	ACM Digital Library

Code	Author	Title	Year	Publication	Source
P5	Choudhary, P.; Kant, V.; Dwivedi, P.	A particle swarm optimization approach to multi criteria recommender system utilizing effective similarity measures	2017	9th International Conference on Machine Learning and Computing	ACM Digital Library
P6	Musto, C.; de Gemmis, M.; Semeraro, G.; Lops, P.	A multi-criteria recommender system exploiting aspect-based sentiment analysis of users' reviews	2017	Eleventh ACM Conference on Recommender Systems	ACM Digital Library
P7	Park, Y.	Recommending personalized tips on new courses for guiding course selection	2017	South-East Conference	ACM Digital Library
P8	Sreepada, R. S.; Patra, B. K.; Hernando, A.	Multi-criteria recommendations through preference learning	2017	Fourth ACM IKDD Conferences on Data Sciences	ACM Digital Library
P9	Zheng, Y.	Criteria chains: A novel multi-criteria recommendation approach	2017	22nd International Conference on Intelligent User Interfaces	ACM Digital Library
P10	Tallapally, D.; Sreepada, R. S.; Patra, B. K.; Babu, K. S.	User preference learning in multi-criteria recommendations using stacked auto encoders	2018	12th ACM Conference on Recommender Systems	ACM Digital Library
P11	Zheng, Y.; Dave, T.; Mishra, N.; Kumar, H.	Fairness in reciprocal recommendations	2018	26th Conference on User Modeling, Adaptation and Personalization	ACM Digital Library
P12	Niknafs, A.; Charkari, N. M.; Niknafs, A. A.	PROMETHEE-based recommender system for multi-sort recommendations in on-line stores	2008	Third International Conference on Digital Information Management	IEEE Xplore

Code	Author	Title	Year	Publication	Source
P13	Hwang, C.-S.; Kao, Y.-C.; Yu, P.	Integrating multiple linear regression and multicriteria collaborative filtering for better recommendation	2010	International Conference on Computational Aspects of Social Networks	IEEE Xplore
P14	Liu, L.; Lecue, F.; Mehandjiev, N.; Xu, L.	Using context similarity for service recommendation	2010	IEEE Fourth International Conference on Semantic Computing	IEEE Xplore
P15	Shambour, Q.; Lu, J.	A framework of hybrid recommendation system for government-to-business personalized e-services	2010	Seventh International Conference on Information Technology: New Generations	IEEE Xplore
P16	Zarrinkalam, F.; Kahani, M.	A multi-criteria hybrid citation recommendation system based on linked data	2012	2nd International eConference on Computer and Knowledge Engineering	IEEE Xplore
P17	Boulkrinat, S.; Hadjali, A.; Mokhtari, A.	Enhancing recommender systems prediction through qualitative preference relations	2013	11th International Symposium on Programming and Systems	IEEE Xplore
P18	Samatthiyadikun, P.; Takasu, A.; Maneeroj, S.	Bayesian model for a multicriteria recommender system with support vector regression	2013	IEEE 14th International Conference on Information Reuse & Integration	IEEE Xplore
P19	Bokde, D. K.; Girase, S.; Mukhopadhyay, D.	An approach to a university recommendation by multi-criteria collaborative filtering and dimensionality reduction techniques	2015	IEEE International Symposium on Nanoelectronic and Information Systems	IEEE Xplore

Code	Author	Title	Year	Publication	Source
P20	Sharma, Y.; Bhatt, J.; Magon, R.	A multi-criteria review-based hotel recommendation system	2015	IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing	IEEE Xplore
P21	Asawarangsee, T.; Maneeroj, S.	A novel aggregation technique for multi-criteria recommendation	2016	13th International Joint Conference on Computer Science and Software Engineering	IEEE Xplore
P22	Ashley-Dejo, E.; Ngwira, S. M.; Zuva, T.	A context-aware proactive recommender system for tourist	2016	International Conference on Advances in Computing and Communication Engineering	IEEE Xplore
P23	Hassan, M.; Hamada, M.	Enhancing learning objects recommendation using multi-criteria recommender systems	2016	IEEE International Conference on Teaching, Assessment, and Learning for Engineering	IEEE Xplore
P24	Wijayanto, A.; Winarko, E.	Implementation of multi-criteria collaborative filtering on cluster using Apache Spark	2016	2nd International Conference on Science and Technology-Computer	IEEE Xplore
P25	Hamada, M.; Odu, N. B.; Hassan, M.	A fuzzy-based approach for modelling preferences of users in multi-criteria recommender systems	2018	IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip	IEEE Xplore

Code	Author	Title	Year	Publication	Source
P26	Mohamed, H.; Abdulsalam, L.; Mohammed, H.	Adaptive genetic algorithm for improving prediction accuracy of a multi-criteria recommender system	2018	IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip	IEEE Xplore
P27	Turk, A. M.; Bilge, A.	A robust multi-criteria collaborative filtering algorithm	2018	Innovations in Intelligent Systems and Applications	IEEE Xplore
P28	Manouselis, N.; Costopoulou, C.	Experimental analysis of design choices in multiattribute utility collaborative filtering	2007	International Journal of Pattern Recognition and Artificial Intelligence	ISI Web of Knowledge
P29	Yin, Z.; Yueting, Z.; Jiangqin, W.; Liang, Z.	Applying probabilistic latent semantic analysis to multi-criteria recommender system	2009	AI Communications	ISI Web of Knowledge
P30	Palanivel, K.; Sivakumar, R.	A study on implicit feedback in multicriteria e-commerce recommender system	2010	Journal of Electronic Commerce Research	ISI Web of Knowledge
P31	Dixit, V. S.; Mehta, H.; Bedi, P.	A proposed framework for group-based multi-criteria recommendations	2014	Applied Artificial Intelligence	ISI Web of Knowledge
P32	Mikeli, A.; Apostolou, D.; Despotis, D.	A new recommendation technique for interval scaled multi-criteria rating systems incorporating intensity of preferences	2015	Intelligent Decision Technologies	ISI Web of Knowledge

Code	Author	Title	Year	Publication	Source
P33	Hamada, M.; Hassan, M.	Artificial neural networks and particle swarm optimization algorithms for preference prediction in multi-criteria recommender systems	2018	Informatics	ISI Web of Knowledge
P34	Chen, D.-N.; Hu, P. J.-H.; Kuo, Y.-R.; Liang, T.-P.	A web-based personalized recommendation system for mobile phone selection: Design, implementation, and evaluation	2010	Expert Systems with Applications	ScienceDirect
P35	Huang, S.-l.	Designing utility-based recommender systems for e-commerce: Evaluation of preference-elicitation methods	2011	Electronic Commerce Research and Applications	ScienceDirect
P36	Liu, H.; He, J.; Wang, T.; Song, W.; Du, X.	Combining user preferences and user opinions for accurate recommendation	2013	Electronic Commerce Research and Applications	ScienceDirect
P37	Hu, Y.-C.	Nonadditive similarity-based single-layer perceptron for multi-criteria collaborative filtering	2014	Neurocomputing	ScienceDirect
P38	Li, Y.-M.; Chou, C.-L.; Lin, L.-F.	A social recommender mechanism for location-based group commerce	2014	Information Sciences	ScienceDirect
P39	Marin, L.; Moreno, A.; Isern, D.	Automatic preference learning on numeric and multi-valued categorical attributes	2014	Knowledge-Based Systems	ScienceDirect

Code	Author	Title	Year	Publication	Source
P40	Nilashi, M.; bin Ibrahim, O.; Ithnin, N.	Hybrid recommendation approaches for multi-criteria collaborative filtering	2014	Expert Systems with Applications	ScienceDirect
P41	Son, L. H.; Thong, N. T.	Intuitionistic fuzzy recommender systems: An effective tool for medical diagnosis	2015	Knowledge-Based Systems	ScienceDirect
P42	Ali, M.; Son, L. H.; Thanh, N. D.; Minh, N. V.	A neutrosophic recommender system for medical diagnosis based on algebraic neutrosophic measures	2017	Applied Soft Computing	ScienceDirect
P43	Amoretti, M.; Belli, L.; Zanichelli, F.	UTravel: Smart mobility with a novel user profiling and recommendation approach	2017	Pervasive and Mobile Computing	ScienceDirect
P44	Choudhary, P.; Kant, V.; Dwivedi, P.	Handling natural noise in multi criteria recommender system utilizing effective similarity measure and particle swarm optimization	2017	Procedia Computer Science	ScienceDirect
P45	Kermany, N. R.; Alizadeh, S. H.	A hybrid multi-criteria recommender system using ontology and neuro-fuzzy techniques	2017	Electronic Commerce Research and Applications	ScienceDirect
P46	Yuen, K. K. F.	The fuzzy cognitive pairwise comparisons for ranking and grade clustering to build a recommender system: An application of smartphone recommendation	2017	Engineering Applications of Artificial Intelligence	ScienceDirect

Code	Author	Title	Year	Publication	Source
P47	Akcayol, M. A.; Utku, A.; Aydoğan, E.; Mutlu, B.	A weighted multi-attribute-based recommender system using extended user behavior analysis	2018	Electronic Commerce Research and Applications	ScienceDirect
P48	Castillo, A.; Meer, D. V.; Castellanos, A.	ExUP recommendations: Inferring user's product metadata preferences from single-criterion rating systems	2018	Decision Support Systems	ScienceDirect
P49	Nilashi, M.; Ibrahim, O.; Yadegaridehkordi, E.; Samad, S.; Akbari, E.; Alizadeh, A.	Travelers decision making using online review in social network sites: A case on TripAdvisor	2018	Journal of Computational Science	ScienceDirect
P50	Núñez-Valdez, E. R.; Quintana, D.; Crespo, R. G.; Isasi, P.; Herrera-Viedma, E.	A recommender system based on implicit feedback for selective dissemination of ebooks	2018	Information Sciences	ScienceDirect
P51	Song, W.; Sakao, T.	An environmentally conscious PSS recommendation method based on users' vague ratings: A rough multi-criteria approach	2018	Journal of Cleaner Production	ScienceDirect
P52	Wasid, M.; Ali, R.	An improved recommender system based on multi-criteria clustering approach	2018	Procedia Computer Science	ScienceDirect
P53	Adomavicius, G.; Kwon, Y.	New recommendation techniques for multicriteria rating systems	2007	IEEE Intelligent Systems	Scopus

Code	Author	Title	Year	Publication	Source
P54	Lee, H.-H.; Teng, W.-G.	Incorporating multi-criteria ratings in recommendation systems	2007	IEEE International Conference on Information Reuse and Integration	Scopus
P55	Hwang, C.-S.	Genetic algorithms for feature weighting in multi-criteria recommender systems	2010	Journal of Convergence Information Technology	Scopus
P56	Lousame, F. P.; Sanchez, E.	Multicriteria predictors using aggregation functions based on item views	2010	10th International Conference on Intelligent Systems Design and Applications	Scopus
P57	Tangphoklang, P.; Maneeroj, S.; Takasu, A.	Advanced representative and dynamic user profile based on MCDM for multi-criteria RS	2010	IADIS International Conference Information Systems	Scopus
P58	Akhtarzada, A.; Calude, C. S.; Hosking, J.	A multi-criteria metric algorithm for recommender systems	2011	Fundamenta Informaticae	Scopus
P59	Lakiotaki, K.; Matsatsinis, N. F.; Tsoukias, A.	Multicriteria user modeling in recommender systems	2011	IEEE Intelligent Systems	Scopus
P60	Palanivel, K.; Sivakumar, R.	A study on collaborative recommender system using fuzzy-multicriteria approaches	2011	International Journal of Business Information Systems	Scopus
P61	Shambour, Q.; Lu, J.	Integrating multi-criteria collaborative filtering and trust filtering for personalized recommender systems	2011	IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making	Scopus

Code	Author	Title	Year	Publication	Source
P62	Samatthiyadikun, P.; Takasu, A.; Maneeroj, S.	Multicriteria collaborative filtering by Bayesian model-based user profiling	2012	IEEE 13th International Conference on Information Reuse & Integration	Scopus
P63	Premchaiswadi, W.; Poompuang, P.	Hybrid profiling for hybrid multicriteria recommendation based on implicit multicriteria information	2013	Applied Artificial Intelligence	Scopus
P64	Agathokleous, M.; Tsapatsoulis, N.	Learning user models in multi-criteria recommender systems	2014	Engineering Applications of Neural Networks	Scopus
P65	Bilge, A.; Kaleli, C.	A multi-criteria item-based collaborative filtering framework	2014	11th International Joint Conference on Computer Science and Software Engineering	Scopus
P66	Hdioud, F.; Frikh, B.; Ouhbi, B.	Bootstrapping recommender systems based on a multi-criteria decision making approach	2014	International Conference on Next Generation Networks and Services	Scopus
P67	Hu, Y.-C.	A multicriteria collaborative filtering approach using the indifference relation and its application to initiator recommendation for group-buying	2014	Applied Artificial Intelligence	Scopus
P68	Manouselis, N.; Kyrgiazos, G.; Stoitsis, G.	Exploratory study of multi-criteria recommendation algorithms over technology enhanced learning datasets	2014	Journal of E-Learning and Knowledge Society	Scopus

Code	Author	Title	Year	Publication	Source
P69	Pinandito, A.; Ananta, M. T.; Brata, K. C.; Fanani, L.	Alternatives weighting in analytic hierarchy process of mobile culinary recommendation system using fuzzy	2015	ARPN Journal of Engineering and Applied Sciences	Scopus
P70	Sneha, Y. S.; Mahadevan, G.	A novel approach to personalized recommender systems based on multi criteria ratings	2015	Research Journal of Applied Sciences, Engineering and Technology	Scopus
P71	Bankshinategh, B.; Spanakis, G.; Zaiane, O.; ElAtia, S.	A course recommender system based on graduating attributes	2017	9th International Conference on Computer Supported Education	Scopus
P72	Goswami, A.; Dwivedi, P.; Kant, V.	Trust-enhanced multi-criteria recommender system	2017	Advances in Intelligent Systems and Computing	Scopus
P73	Hassan, M.; Hamada, M.	A neural networks approach for improving the accuracy of multi-criteria recommender systems	2017	Applied Sciences	Scopus
P74	Leal, F.; González-Vélez, H.; Malheiro, B.; Burguillo, J. C.	Profiling and rating prediction from multi-criteria crowd-sourced hotel ratings	2017	31st European Conference on Modelling and Simulation	Scopus
P75	Majumder, G. S.; Dwivedi, P.; Kant, V.	Matrix factorization and regression-based approach for multi-criteria recommender system	2017	Information and Communication Technology for Intelligent Systems	Scopus
P76	Karacapilidis, N.; Hatzieleftheriou, L.	Exploiting similarity measures in multi-criteria based recommendations	2003	E-Commerce and Web Technologies	Springer Link

Code	Author	Title	Year	Publication	Source
P77	Manouselis, N.; Costopoulou, C.	Preliminary study of the expected performance of MAUT collaborative filtering algorithms	2008	The Open Knowlege Society	Springer Link
P78	Matsatsinis, N. F.; Manarolis, E. A.	New hybrid recommender approaches: An application to equity funds selection	2009	Algorithmic Decision Theory	Springer Link
P79	Naak, A.; Hage, H.; Aïmeur, E.	A multi-criteria collaborative filtering approach for research paper recommendation in Papyres	2009	E-Technologies: Innovation in an Open World	Springer Link
P80	Bitonto, P. D.; Laterza, M.; Roselli, T.; Rossano, V.	Multi-criteria retrieval in cultural heritage recommendation systems	2010	Knowledge-Based and Intelligent Information and Engineering Systems	Springer Link
P81	Maneeroj, S.; Samatthiyadikun, P.; Chalermpornpong, W.; Panthuwadeethorn, S.; Takasu, A.	Ranked criteria profile for multi-criteria rating recommender	2012	Information Systems, Technology and Management	Springer Link
P82	Fan, J.; Xu, L.	A robust multi-criteria recommendation approach with preference-based similarity and support vector machine	2013	Advances in Neural Networks	Springer Link
P83	Jannach, D.; Zanker, M.; Fuchs, M.	Leveraging multi-criteria customer feedback for satisfaction analysis and improved recommendations	2014	Information Technology & Tourism	Springer Link

Code	Author	Title	Year	Publication	Source
P84	Nilashi, M.; Ibrahim, O. B.; Ithnin, N.; Zakaria, R.	A multi-criteria recommendation system using dimensionality reduction and neuro-fuzzy techniques	2014	Soft Computing	Springer Link
P85	Chen, T.; Chuang, Y. H.	Fuzzy and nonlinear programming approach for optimizing the performance of ubiquitous hotel recommendation	2015	Journal of Ambient Intelligence and Humanized Computing	Springer Link
P86	Li, S. T.; Pham, T. T.; Chuang, H. C.; Wang, Z.-W.	Does reliable information matter? Towards a trustworthy co-created recommendation model by mining unboxing reviews	2015	Information Systems and e-Business Management	Springer Link
P87	Parveen, R.; Kant, V.; Dwivedi, P.; Jaiswal, A. K.	Enhancing recommendation quality of a multi criterion recommender system using genetic algorithm	2015	Mining Intelligence and Knowledge Exploration	Springer Link
P88	Jhalani, T.; Kant, V.; Dwivedi, P.	A linear regression approach to multi-criteria recommender system	2016	Data Mining and Big Data	Springer Link
P89	Kant, V.; Jhalani, T.; Dwivedi, P.	Enhanced multi-criteria recommender system based on fuzzy Bayesian approach	2017	Multimedia Tools and Applications	Springer Link
P90	Ko, H.-G.; Ko, I.-Y.; Lee, D.	Multi-criteria matrix localization and integration for personalized collaborative filtering in IoT environments	2017	Multimedia Tools and Applications	Springer Link

Code	Author	Title	Year	Publication	Source
P91	Leal, F.; Malheiro, B.; González-Vélez, H.; Burguillo, J. C.	Trust-based modelling of multi-criteria crowdsourced data	2017	Data Science and Engineering	Springer Link
P92	Ding, Y.; Li, S.; Yu, W.	Multi-criteria recommendation schemes based on factorization machines	2018	Cluster Computing	Springer Link
P93	Ding, Y.; Li, S.; Yu, W.; Wang, J.; Liu, M.	A unified neural model for review-based rating prediction by leveraging multi-criteria ratings and review text	2018	Cluster Computing	Springer Link

Appendix B

Publication List

The studies discussed as part of the present dissertation, in which I have been either the main author or a co-author, have been published in the following conference proceedings or journals.

- Iacopo Vagliano, Diego Monti, and Maurizio Morisio. "SemRevRec: A Recommender System based on User Reviews and Linked Data". In: *Proceedings of the Poster Track of the 11th ACM Conference on Recommender Systems*. Como, Italy: CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-1905/recsys2017_poster10.pdf

- Iacopo Vagliano, Diego Monti, Ansgar Scherp, and Maurizio Morisio. "Content Recommendation Through Semantic Annotation of User Reviews and Linked Data". In: *Proceedings of the Knowledge Capture Conference*. K-CAP 2017. Austin, TX, USA: ACM, 2017, 32:1–32:4. DOI: 10.1145/3148011.3148035

- Diego Monti, Enrico Palumbo, Giuseppe Rizzo, and Maurizio Morisio. "Sequeval: A framework to assess and benchmark sequence-based recommender systems". In: *Proceedings of the Workshop on Offline Evaluation for Recommender Systems co-located with the 12th ACM Conference on Recommender Systems*. Vancouver, BC, Canada: Workshop Organizers, 2018. URL: https://arxiv.org/abs/1810.04956

- Diego Monti, Giuseppe Rizzo, and Maurizio Morisio. "A Distributed and Accountable Approach to Offline Recommender Systems Evaluation". In: *Proceedings of the Workshop on Offline Evaluation for Recommender Systems co-located with the 12th ACM Conference on Recommender Systems*. Vancouver, BC, Canada: Workshop Organizers, 2018. URL: https://arxiv.org/abs/1810.04957

- Diego Monti, Enrico Palumbo, Giuseppe Rizzo, Pasquale Lisena, Raphaël Troncy, Michael Fell, Elena Cabrio, and Maurizio Morisio. "An Ensemble Approach of Recurrent Neural Networks Using Pre-Trained Embeddings for Playlist Completion". In: *Proceedings of the ACM Recommender Systems Challenge 2018*. RecSys Challenge '18. Vancouver, BC, Canada: ACM, 2018, 13:1–13:6. DOI: 10.1145/3267471.3267484

- Diego Monti, Enrico Palumbo, Giuseppe Rizzo, and Maurizio Morisio. "Sequeval: An Offline Evaluation Framework for Sequence-Based Recommender Systems". In: *Information* 10.5 (May 2019), p. 174. DOI: 10.3390/info10050174

- Diego Monti, Giuseppe Rizzo, and Maurizio Morisio. "Visualizing Ratings in Recommender System Datasets". In: *Proceedings of the 6th Joint Workshop on Interfaces and Human Decision Making for Recommender Systems co-located with the 13th ACM Conference on Recommender Systems*. Copenhagen, Denmark: CEUR-WS.org, 2019, pp. 60–64. URL: http://ceur-ws.org/Vol-2450/short2.pdf

- Diego Monti, Giuseppe Rizzo, and Maurizio Morisio. "All You Need is Ratings: A Clustering Approach to Synthetic Rating Datasets Generation". In: *Proceedings of the Workshop on Reinforcement and Robust Estimators co-located with the 13th ACM Conference on Recommender Systems*. Copenhagen, Denmark: Workshop Organizers, 2019. URL: https://arxiv.org/abs/1909.00687

Furthermore, in the context of my Ph.D. career, I have collaborated as a co-author to the realization of the following studies.

- Giulio Carducci, Giuseppe Rizzo, Diego Monti, Enrico Palumbo, and Maurizio Morisio. "TwitPersonality: Computing Personality Traits from Tweets Using Word Embeddings and Supervised Learning". In: *Information* 9.5 (May 2018), p. 127. DOI: 10.3390/info9050127

- Simone Leonardi, Diego Monti, Giuseppe Rizzo, and Maurizio Morisio. "Mining Micro-Influencers from Social Media Posts". In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. Brno, Czech Republic: ACM, 2020, pp. 867–874. DOI: 10.1145/3341105.3373954

- Simone Leonardi, Diego Monti, Giuseppe Rizzo, and Maurizio Morisio. "Multilingual Transformer-Based Personality Traits Estimation". In: *Information* 11.4 (Apr. 2020), p. 179. DOI: 10.3390/info11040179

- Enrico Palumbo, Diego Monti, Giuseppe Rizzo, Raphaël Troncy, and Elena Baralis. "entity2rec: Property-Specific Knowledge Graph Embeddings for Item Recommendation". In: *Expert Systems with Applications* 151 (Aug. 2020), p. 113235. DOI: 10.1016/j.eswa.2020.113235

The quality of the conferences and journals selected as publication venues of the aforementioned studies is attested by the following indicators.

- Expert Systems with Applications, CiteScore quartile: Q1
- Information, CiteScore quartile: Q2
- Knowledge Capture Conference, CORE rank: A
- ACM Symposium on Applied Computing, CORE rank: B
- ACM International Conference on Recommender Systems, CORE rank: B

Bibliography

[1] Silvana Aciar, Debbie Zhang, Simeon Simoff, and John Debenham. "Informed Recommender: Basing Recommendations on Consumer Product Reviews". In: *IEEE Intelligent Systems* 22.3 (2007), pp. 39–47. ISSN: 1541-1672. DOI: 10.1109/MIS.2007.55.

[2] Gediminas Adomavicius and YoungOk Kwon. "Multi-Criteria Recommender Systems". In: *Recommender Systems Handbook*. Springer, 2015, pp. 847–880. DOI: 10.1007/978-1-4899-7637-6_25.

[3] Gediminas Adomavicius and Alexander Tuzhilin. "Context-Aware Recommender Systems". In: *Recommender Systems Handbook*. Springer, 2015, pp. 191–226. DOI: 10.1007/978-1-4899-7637-6_6.

[4] Gediminas Adomavicius and Alexander Tuzhilin. "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions". In: *IEEE Transactions on Knowledge and Data Engineering* 17.6 (2005), pp. 734–749. DOI: 10.1109/tkde.2005.99.

[5] Charu C. Aggarwal. "Mining Discrete Sequences". In: *Data Mining*. Springer International Publishing, 2015. Chap. 15, pp. 493–529. DOI: 10.1007/978-3-319-14142-8_15.

[6] Rakesh Agrawal and Ramakrishnan Srikant. "Fast Algorithms for Mining Association Rules in Large Databases". In: *Proceedings of the 20th International Conference on Very Large Data Bases*. VLDB '94. Burlington, MA, USA: Morgan Kaufmann Publishers, 1994, pp. 487–499.

[7] Massimiliano Albanese, Angelo Chianese, Antonio d'Acierno, Vincenzo Moscato, and Antonio Picariello. "A multimedia recommender integrating object features and user behavior". In: *Multimedia Tools and Applications* 50.3 (2010), pp. 563–585. DOI: 10.1007/s11042-010-0480-8.

[8] Massimiliano Albanese, Antonio d'Acierno, Vincenzo Moscato, Fabio Persia, and Antonio Picariello. "A Multimedia Recommender System". In: *ACM Transactions on Internet Technology* 13.1 (2013), pp. 1–32. DOI: 10.1145/2532640.

[9] Massimiliano Albanese, Antonio d'Acierno, Vincenzo Moscato, Fabio Persia, and Antonio Picariello. "A Multimedia Semantic Recommender System for Cultural Heritage Applications". In: *Fifth International Conference on Semantic Computing*. IEEE, 2011. DOI: 10.1109/icsc.2011.47.

[10] Flora Amato, Vincenzo Moscato, Antonio Picariello, and Giancarlo Sperli. "KIRA: A System for Knowledge-Based Access to Multimedia Art Collections". In: *11th International Conference on Semantic Computing*. IEEE, 2017. DOI: 10.1109/icsc.2017.59.

[11] Marko Balabanović and Yoav Shoham. "Fab: content-based, collaborative recommendation". In: *Communications of the ACM* 40.3 (1997), pp. 66–72. DOI: 10.1145/245108.245124.

[12] Chumki Basu, Haym Hirsh, and William Cohen. "Recommendation As Classification: Using Social and Content-based Information in Recommendation". In: *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. Madison, Wisconsin, USA: AAAI Press, 1998, pp. 714–720. URL: https://www.aaai.org/Papers/AAAI/1998/AAAI98-101.pdf.

[13] Alejandro Bellogín, Iván Cantador, and Pablo Castells. "A Study of Heterogeneity in Recommendations for a Social Music Service". In: *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*. HetRec '10. ACM, 2010, pp. 1–8. ISBN: 978-1-4503-0407-8. DOI: 10.1145/1869446.1869447.

[14] Alejandro Bellogín, Pablo Castells, and Iván Cantador. "Statistical biases in Information Retrieval metrics for recommender systems". In: *Information Retrieval Journal* 20.6 (2017), pp. 606–634. DOI: 10.1007/s10791-017-9312-z.

[15] Alejandro Bellogín and Pablo Sánchez. "Collaborative filtering based on subsequence matching: A new approach". In: *Information Sciences* 418-419 (2017), pp. 432–446. DOI: 10.1016/j.ins.2017.08.016.

[16] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* 3 (2003), pp. 1137–1155. URL: http://www.jmlr.org/papers/v3/bengio03a.html.

[17] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation". In: *Journal of Machine Learning Research* 3 (2003), pp. 993–1022. ISSN: 1532-4435.

[18] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. "Recommender systems survey". In: *Knowledge-Based Systems* 46 (July 2013), pp. 109–132. DOI: 10.1016/j.knosys.2013.03.012.

[19] Robin Burke. "Hybrid Web Recommender Systems". In: *The Adaptive Web*. Springer Berlin Heidelberg, 2007, pp. 377–408. DOI: 10.1007/978-3-540-72079-9_12.

[20] Pedro G. Campos, Fernando Díez, and Iván Cantador. "Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols". In: *User Modeling and User-Adapted Interaction* 24.1-2 (2013), pp. 67–119. DOI: 10.1007/s11257-012-9136-x.

[21] Erion Çano and Maurizio Morisio. "Hybrid recommender systems: A systematic literature review". In: *Intelligent Data Analysis* 21.6 (2017), pp. 1487–1524. ISSN: 1088-467X. DOI: 10.3233/IDA-163209.

[22] Iván Cantador, Peter Brusilovsky, and Tsvi Kuflik. "Second Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec2011)". In: *Proceedings of the Fifth ACM Conference on Recommender Systems*. RecSys '11. Chicago, Illinois, USA: ACM, 2011, pp. 387–388. ISBN: 978-1-4503-0683-6. DOI: 10.1145/2043932.2044016.

[23] Bruno Cardoso, Gayane Sedrakyan, Francisco Gutiérrez, Denis Parra, Peter Brusilovsky, and Katrien Verbert. "IntersectionExplorer, a multi-perspective approach for exploring recommendations". In: *International Journal of Human-Computer Studies* 121 (2019), pp. 73–92. DOI: 10.1016/j.ijhcs.2018.04.008.

[24] Giulio Carducci, Giuseppe Rizzo, Diego Monti, Enrico Palumbo, and Maurizio Morisio. "TwitPersonality: Computing Personality Traits from Tweets Using Word Embeddings and Supervised Learning". In: *Information* 9.5 (May 2018), p. 127. DOI: 10.3390/info9050127.

[25] María del Carmen Rodríguez-Hernández, Sergio Ilarri, Ramón Hermoso, and Raquel Trillo-Lado. "DataGenCARS: A generator of synthetic data for the evaluation of context-aware recommendation systems". In: *Pervasive and Mobile Computing* 38 (2017), pp. 516–541. DOI: 10.1016/j.pmcj.2016.09.020.

[26] Patrali Chatterjee. "Online reviews: Do consumers use them?" In: *Advances in Consumer Research* 28 (2001), pp. 129–133. URL: http://acrwebsite.org/volumes/8455/volumes/v28/NA-28.

[27] Li Chen, Guanliang Chen, and Feng Wang. "Recommender systems based on user reviews: The state of the art". In: *User Modeling and User-Adapted Interaction* 25.2 (2015), pp. 99–154. ISSN: 1573-1391. DOI: 10.1007/s11257-015-9155-5.

[28] 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*. ACM Press, 2012, pp. 714–722. DOI: 10.1145/2339530.2339643.

[29] Stanley F. Chen and Joshua Goodman. "An empirical study of smoothing techniques for language modeling". In: *Computer Speech & Language* 13.4 (1999), pp. 359–393. DOI: 10.1006/csla.1999.0128.

[30] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: *arXiv preprint* (2014). URL: https://arxiv.org/abs/1412.3555.

[31] Ludovik Çoba, Panagiotis Symeonidis, and Markus Zanker. "Visual Analysis of Recommendation Performance". In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. RecSys '17. Como, Italy: ACM, 2017, pp. 362–363. ISBN: 978-1-4503-4652-8. DOI: 10.1145/3109859.3109982.

[32] Alberto Costa and Fabio Roda. "Recommender systems by means of information retrieval". In: *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*. ACM Press, 2011. DOI: 10.1145/1988688.1988755.

[33] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. "Performance of Recommender Algorithms on Top-n Recommendation Tasks". In: *Proceedings of the Fourth ACM Conference on Recommender Systems*. RecSys '10. New York, NY, USA: ACM, 2010, pp. 39–46. DOI: 10.1145/1864708.1864721.

[34] Daniela S. Cruzes and Tore Dyba. "Recommended Steps for Thematic Synthesis in Software Engineering". In: *2011 International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2011. DOI: 10.1109/esem.2011.36.

[35] Joachim Daiber, Max Jakob, Chris Hokamp, and Pablo N. Mendes. "Improving Efficiency and Accuracy in Multilingual Entity Extraction". In: *Proceedings of the 9th International Conference on Semantic Systems*. I-SEMANTICS '13. ACM, 2013. DOI: 10.1145/2506182.2506198.

[36] Danica Damljanovic, Milan Stankovic, and Philippe Laublet. "Linked Data-Based Concept Recommendation: Comparison of Different Methods in Open Innovation Scenario". In: *The Semantic Web: Research and Applications*. Springer Berlin Heidelberg, 2012, pp. 24–38. ISBN: 978-3-642-30283-1. DOI: 10.1007/978-3-642-30284-8_9.

[37] Tommaso Di Noia and Vito Claudio Ostuni. "Recommender Systems and Linked Open Data". In: *Reasoning Web. Web Logic Rules*. Springer International Publishing, 2015, pp. 88–113. ISBN: 978-3-319-21768-0. DOI: 10.1007/978-3-319-21768-0_4.

[38] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, and Davide Romito. "Exploiting the Web of Data in Model-based Recommender Systems". In: *Proceedings of the Sixth ACM Conference on Recommender Systems*. RecSys '12. Dublin, Ireland: ACM, 2012, pp. 253–256. ISBN: 978-1-4503-1270-7. DOI: 10.1145/2365952.2366007.

[39] Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, and Eugenio Di Sciascio. "SPrank: Semantic Path-Based Ranking for Top-N Recommendations Using Linked Open Data". In: *ACM Trans. Intell. Syst. Technol.* 8.1 (2016), 9:1–9:34. ISSN: 2157-6904. DOI: 10.1145/2899005.

[40] Yi Ding and Xue Li. "Time weight collaborative filtering". In: *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. ACM Press, 2005, pp. 485–492. DOI: 10.1145/1099554.1099689.

[41] Ruihai Dong, Markus Schaal, Michael P. O'Mahony, Kevin McCarthy, and Barry Smyth. "Opinionated Product Recommendation". In: *Case-Based Reasoning Research and Development*. Springer Berlin Heidelberg, 2013, pp. 44–58. DOI: 10.1007/978-3-642-39056-2_4.

[42] Jakub Dzikowski, Monika Kaczmarek, Szymon Lazaruk, and Witold Abramowicz. "Challenges in Using Linked Data within a Social Web Recommendation Application to Semantically Annotate and Discover Venues". In: *Multidisciplinary Research and Practice for Information Systems*. Springer Berlin Heidelberg, 2012, pp. 360–374. ISBN: 978-3-642-32498-7. DOI: 10.1007/978-3-642-32498-7_27.

[43] 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*. Buenos Aires, Argentina: AAAI Press, 2015, pp. 2069–2075. ISBN: 978-1-57735-738-4. URL: https://www.ijcai.org/Proceedings/15/Papers/293.pdf.

[44] Cristhian Figueroa, Iacopo Vagliano, Oscar Rodríguez Rocha, and Maurizio Morisio. "A systematic literature review of Linked Data-based recommender systems". In: *Concurrency and Computation: Practice and Experience* 27.17 (2015), pp. 4659–4684. DOI: 10.1002/cpe.3449.

[45] Aldo Gangemi. "A Comparison of Knowledge Extraction Tools for the Semantic Web". In: *The Semantic Web: Semantics and Big Data*. ESWC 2013. Springer Berlin Heidelberg, 2013, pp. 351–366. DOI: 10.1007/978-3-642-38288-8_24.

[46] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. "MyMediaLite: A Free Recommender System Library". In: *Proceedings of the 5th ACM Conference on Recommender Systems*. RecSys '11. Chicago, Illinois, USA: ACM, 2011, pp. 305–308. DOI: 10.1145/2043932.2043989.

[47] Mouzhi Ge, Carla Delgado-Battenfeld, and Dietmar Jannach. "Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity". In: *Proceedings of the Fourth ACM Conference on Recommender Systems*. RecSys '10. New York, NY, USA: ACM, 2010, pp. 257–260. DOI: 10.1145/1864708.1864761.

[48] Marco de Gemmis, Pasquale Lops, Giovanni Semeraro, and Cataldo Musto. "An investigation on the serendipity problem in recommender systems". In: *Information Processing & Management* 51.5 (2015), pp. 695–717. DOI: 10.1016/j.ipm.2015.06.008.

[49] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. "Semantics-Aware Content-Based Recommender Systems". In: *Recommender Systems Handbook*. Springer, 2015, pp. 119–159. DOI: 10.1007/978-1-4899-7637-6_4.

[50] Stephanie Gil, Jesús Bobadilla, Fernando Ortega, and Bo O. Zhu. "VisualRS: Java framework for visualization of recommender systems information". In: *Knowledge-Based Systems* 155 (2018), pp. 66–70. DOI: 10.1016/j.knosys.2018.04.028.

[51] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. "Using collaborative filtering to weave an information tapestry". In: *Communications of the ACM* 35.12 (1992), pp. 61–70. DOI: 10.1145/138859.138867.

[52] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016. ISBN: 978-0-262-03561-3.

[53] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Nets". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems*. Vol. 2. Cambridge, MA, USA: MIT Press, 2014, pp. 2672–2680.

[54] Alex Graves, Abdel-Rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks". In: *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013. DOI: 10.1109/icassp.2013.6638947.

[55] Asela Gunawardana and Guy Shani. "Evaluating Recommender Systems". In: *Recommender Systems Handbook*. Springer, 2015. Chap. 8, pp. 265–308. DOI: 10.1007/978-1-4899-7637-6_8.

[56] Ido Guy. "Social Recommender Systems". In: *Recommender Systems Handbook*. Springer, 2015, pp. 511–543. DOI: 10.1007/978-1-4899-7637-6_15.

[57] F. Maxwell Harper and Joseph A. Konstan. "The MovieLens Datasets: History and Context". In: *ACM Transactions on Interactive Intelligent Systems* 5.4 (2015), pp. 1–19. DOI: 10.1145/2827872.

[58] John A. Hartigan and Manchek A. Wong. "Algorithm AS 136: A K-Means Clustering Algorithm". In: *Applied Statistics* 28.1 (1979), p. 100. DOI: 10.2307/2346830.

[59] Ruining He, Wang-Cheng Kang, and Julian McAuley. "Translation-based Recommendation". In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM Press, 2017, pp. 161–169. DOI: 10.1145/3109859.3109882.

[60] Benjamin Heitmann and Conor Hayes. "Using Linked Data to Build Open, Collaborative Recommender Systems". In: *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*. AAAI, 2010, pp. 76–81.

[61] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. "Explaining collaborative filtering recommendations". In: *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*. ACM Press, 2000, pp. 241–250. DOI: 10.1145/358916.358995.

[62] 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 (2004), pp. 5–53. DOI: 10.1145/963770.963772.

[63] Daniel S. Hirschberg. "A linear space algorithm for computing maximal common subsequences". In: *Communications of the ACM* 18.6 (1975), pp. 341–343. DOI: 10.1145/360825.360861.

[64] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[65] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. "Robust Disambiguation of Named Entities in Text". In: *EMNLP 2011*. ACL, 2011, pp. 782–792.

[66] Jong-yi Hong, Eui-ho Suh, and Sung-Jin Kim. "Context-aware systems: A literature review and classification". In: *Expert Systems with Applications* 36.4 (2009), pp. 8509–8522. DOI: 10.1016/j.eswa.2008.10.071.

[67] Kenneth Houkjær, Kristian Torp, and Rico Wind. "Simple and Realistic Data Generation". In: *Proceedings of the 32nd International Conference on Very Large Data Bases*. VLDB '06. Seoul, Korea: VLDB Endowment, 2006, pp. 1243–1246.

[68] 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. Washington, DC, USA: IEEE Computer Society, 2008, pp. 263–272. ISBN: 978-0-7695-3502-9. DOI: 10.1109/ICDM.2008.22.

[69] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511763113.

[70] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. "What recommenders recommend: an analysis of recommendation biases and possible countermeasures". In: *User Modeling and User-Adapted Interaction* 25.5 (2015), pp. 427–491. DOI: 10.1007/s11257-015-9165-3.

[71] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated gain-based evaluation of IR techniques". In: *ACM Transactions on Information Systems* 20.4 (2002), pp. 422–446. DOI: 10.1145/582415.582418.

[72] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. "FastText.zip: Compressing text classification models". In: *arXiv preprint* (2016). URL: https://arxiv.org/abs/1612.03651.

[73] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Prentice Hall, 2008. ISBN: 978-0-13-187321-6.

[74] Andrej Karpathy and Li Fei-Fei. "Deep visual-semantic alignments for generating image descriptions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3128–3137. DOI: 10.1109/TPAMI.2016.2598339.

[75] Young Ae Kim and Jaideep Srivastava. "Impact of Social Influence in e-Commerce Decision Making". In: *Proceedings of the Ninth International Conference on Electronic Commerce*. ICEC '07. Minneapolis, MN, USA: ACM, 2007, pp. 293–302. ISBN: 978-1-59593-700-1. DOI: 10.1145/1282100.1282157.

[76] Barbara Kitchenham and Stuart Charters. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007.

[77] Johannes Kunkel, Benedikt Loepp, and Jürgen Ziegler. "A 3D Item Space Visualization for Presenting and Manipulating User Preferences in Collaborative Filtering". In: *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. IUI '17. New York, NY, USA: ACM, 2017, pp. 3–15. DOI: 10.1145/3025171.3025189.

[78] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[79] Simone Leonardi, Diego Monti, Giuseppe Rizzo, and Maurizio Morisio. "Mining Micro-Influencers from Social Media Posts". In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing*. Brno, Czech Republic: ACM, 2020, pp. 867–874. DOI: 10.1145/3341105.3373954.

[80] Simone Leonardi, Diego Monti, Giuseppe Rizzo, and Maurizio Morisio. "Multilingual Transformer-Based Personality Traits Estimation". In: *Information* 11.4 (Apr. 2020), p. 179. DOI: 10.3390/info11040179.

[81] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. "Recommender system application developments: A survey". In: *Decision Support Systems* 74 (June 2015), pp. 12–32. DOI: 10.1016/j.dss.2015.03.008.

[82] 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. DOI: 10.1007/s11257-018-9209-6.

[83] Nikos Manouselis and Constantina Costopoulou. "Analysis and Classification of Multi-Criteria Recommender Systems". In: *World Wide Web* 10.4 (2007), pp. 415–441. DOI: 10.1007/s11280-007-0019-8.

[84] Nikos Manouselis and Constantina Costopoulou. "Preliminary Study of the Expected Performance of MAUT Collaborative Filtering Algorithms". In: *The Open Knowlege Society. A Computer Science and Information Systems Manifesto*. New York, NY, USA: Springer Publishing, 2008, pp. 527–536. DOI: 10.1007/978-3-540-87783-7_67.

[85] Gabriel Meseguer-Brocal, Geoffroy Peeters, Guillaume Pellerin, Michel Buffa, Elena Cabrio, Catherine Faron Zucker, Alain Giboin, Isabelle Mirbel, Romain Hennequin, Manuel Moussallam, Francesco Piccoli, and Thomas Fillon. "WASABI: a Two Million Song Database Project with Audio and Cultural Metadata plus WebAudio enhanced Client Applications". In: *Proceedings of 3rd Web Audio Conference*. London, United Kingdom, 2017. URL: https://qmro.qmul.ac.uk/xmlui/handle/123456789/26123.

[86] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[87] Tomas Mikolov, Martin Karafiat, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. "Recurrent neural network based language model". In: *Interspeech*. Vol. 2. 2010, p. 3.

[88] Miquel Montaner, Beatriz López, and Josep Lluís de la Rosa. "Evaluation of recommender systems through simulated users". In: *Proceedings of the 4th International Workshop on Pattern Recognition in Information Systems*. Setúbal, Portugal: SciTePress, 2004, pp. 1–6. DOI: 10.5220/0002622703030308.

[89] Diego Monti, Giuseppe Rizzo, and Maurizio Morisio. "A Distributed and Accountable Approach to Offline Recommender Systems Evaluation". In: *Proceedings of the Workshop on Offline Evaluation for Recommender Systems co-located with the 12th ACM Conference on Recommender Systems*. Vancouver, BC, Canada: Workshop Organizers, 2018. URL: https://arxiv.org/abs/1810.04957.

[90] Diego Monti, Giuseppe Rizzo, and Maurizio Morisio. "All You Need is Ratings: A Clustering Approach to Synthetic Rating Datasets Generation". In: *Proceedings of the Workshop on Reinforcement and Robust Estimators co-located with the 13th ACM Conference on Recommender Systems*. Copenhagen, Denmark: Workshop Organizers, 2019. URL: https://arxiv.org/abs/1909.00687.

[91] Diego Monti, Giuseppe Rizzo, and Maurizio Morisio. "Visualizing Ratings in Recommender System Datasets". In: *Proceedings of the 6th Joint Workshop on Interfaces and Human Decision Making for Recommender Systems co-located with the 13th ACM Conference on Recommender Systems*. Copenhagen, Denmark: CEUR-WS.org, 2019, pp. 60–64. URL: http://ceur-ws.org/Vol-2450/short2.pdf.

[92] Diego Monti, Enrico Palumbo, Giuseppe Rizzo, Pasquale Lisena, Raphaël Troncy, Michael Fell, Elena Cabrio, and Maurizio Morisio. "An Ensemble Approach of Recurrent Neural Networks Using Pre-Trained Embeddings for Playlist Completion". In: *Proceedings of the ACM Recommender Systems Challenge 2018*. RecSys Challenge '18. Vancouver, BC, Canada: ACM, 2018, 13:1–13:6. DOI: 10.1145/3267471.3267484.

[93] Diego Monti, Enrico Palumbo, Giuseppe Rizzo, and Maurizio Morisio. "Sequeval: A framework to assess and benchmark sequence-based recommender systems". In: *Proceedings of the Workshop on Offline Evaluation for Recommender Systems co-located with the 12th ACM Conference on Recommender*

Systems. Vancouver, BC, Canada: Workshop Organizers, 2018. URL: https://arxiv.org/abs/1810.04956.

[94] Diego Monti, Enrico Palumbo, Giuseppe Rizzo, and Maurizio Morisio. "Sequeval: An Offline Evaluation Framework for Sequence-Based Recommender Systems". In: *Information* 10.5 (May 2019), p. 174. DOI: 10.3390/info10050174.

[95] Cataldo Musto, Pasquale Lops, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. "Semantics-aware Graph-based Recommender Systems Exploiting Linked Open Data". In: *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*. UMAP '16. Halifax, Nova Scotia, Canada: ACM, 2016, pp. 229–237. ISBN: 978-1-4503-4368-8. DOI: 10.1145/2930238.2930249.

[96] Jorge Nocedal. "Updating quasi-Newton matrices with limited storage". In: *Mathematics of Computation* 35.151 (1980), pp. 773–782. DOI: 10.1090/s0025-5718-1980-0572855-7.

[97] Tommaso Di Noia, Vito Claudio Ostuni, Jessica Rosati, Paolo Tomeo, and Eugenio Di Sciascio. "An analysis of users' propensity toward diversity in recommendations". In: *Proceedings of the 8th ACM Conference on Recommender Systems*. ACM Press, 2014, pp. 285–288. DOI: 10.1145/2645710.2645774.

[98] Vito Claudio Ostuni, Tommaso Di Noia, Roberto Mirizzi, and Eugenio Di Sciascio. "A Linked Data Recommender System Using a Neighborhood-Based Graph Kernel". In: *E-Commerce and Web Technologies*. EC-Web 2014. Springer International Publishing, 2014, pp. 89–100. ISBN: 978-3-319-10491-1. DOI: 10.1007/978-3-319-10491-1_10.

[99] Enrico Palumbo, Diego Monti, Giuseppe Rizzo, Raphaël Troncy, and Elena Baralis. "entity2rec: Property-Specific Knowledge Graph Embeddings for Item Recommendation". In: *Expert Systems with Applications* 151 (Aug. 2020), p. 113235. DOI: 10.1016/j.eswa.2020.113235.

[100] Enrico Palumbo, Giuseppe Rizzo, Raphaël Troncy, and Elena Baralis. "Predicting Your Next Stop-over from Location-based Social Network Data with Recurrent Neural Networks". In: *Proceedings of the 2nd Workshop on Recommenders in Tourism co-located with 11th ACM Conference on Recommender Systems*. CEUR Workshop Proceedings 1906. CEUR-WS.org, 2017, pp. 1–8. URL: http://ceur-ws.org/Vol-1906/paper1.pdf.

[101] Dimitris Paraschakis, Bengt J. Nilsson, and John Hollander. "Comparative Evaluation of Top-N Recommenders in e-Commerce: An Industrial Perspective". In: *IEEE 14th International Conference on Machine Learning and Applications*. IEEE, 2015, pp. 1024–1031. DOI: 10.1109/icmla.2015.183.

[102] Deuk Hee Park, Hyea Kyeong Kim, Il Young Choi, and Jae Kyeong Kim. "A literature review and classification of recommender systems research". In: *Expert Systems with Applications* 39.11 (2012), pp. 10059–10072. DOI: 10.1016/j.eswa.2012.02.038.

[103] Marden Pasinato, Carlos Eduardo Mello, Marie-Aude Aufaure, and Geraldo Zimbrao. "Generating Synthetic Data for Context-Aware Recommender Systems". In: *2013 BRICS Congress on Computational Intelligence and 11th Brazilian Congress on Computational Intelligence*. New York, NY, USA: IEEE, 2013, pp. 563–567. DOI: 10.1109/brics-cci-cbic.2013.99.

[104] Alexandre Passant. "dbrec: Music Recommendations Using DBpedia". In: *The Semantic Web*. Vol. 2. ISWC 2010. Springer Berlin Heidelberg, 2010, pp. 209–224. DOI: 10.1007/978-3-642-17749-1_14.

[105] Ivens Portugal, Paulo Alencar, and Donald Cowan. "The use of machine learning algorithms in recommender systems: A systematic review". In: *Expert Systems with Applications* 97 (2018), pp. 205–227. DOI: 10.1016/j.eswa.2017.12.020.

[106] Vasiliki Pouli, Stella Kafetzoglou, Eirini Eleni Tsiropoulou, Aggeliki Dimitriou, and Symeon Papavassiliou. "Personalized multimedia content retrieval through relevance feedback techniques for enhanced user experience". In: *13th International Conference on Telecommunications*. IEEE, 2015. DOI: 10.1109/contel.2015.7231205.

[107] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. "Opinion Word Expansion and Target Extraction through Double Propagation". In: *Comput. Linguist.* 37.1 (2011), pp. 9–27. ISSN: 0891-2017. DOI: 10.1162/coli_a_00034.

[108] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. "Sequence-Aware Recommender Systems". In: *ACM Computing Surveys* 51.4 (2018), pp. 1–36. DOI: 10.1145/3190616.

[109] Radim Rehurek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, 2010, pp. 45–50.

[110] 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*. ACM Press, 2010, pp. 811–820. DOI: 10.1145/1772690.1772773.

[111] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. "BPR: Bayesian Personalized Ranking from Implicit Feedback". In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. UAI '09. Montreal, Quebec, Canada: AUAI Press, 2009, pp. 452–

461. ISBN: 978-0-9749039-5-8. URL: https://arxiv.org/pdf/1205.2618.pdf.

[112] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Recommender Systems: Introduction and Challenges". In: *Recommender Systems Handbook*. Springer, 2015. Chap. 1, pp. 1–34. DOI: 10.1007/978-1-4899-7637-6_1.

[113] Cornelis Joost van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 1979.

[114] Graeme D. Ruxton. "The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test". In: *Behavioral Ecology* 17.4 (2006), pp. 688–690. DOI: 10.1093/beheco/ark016.

[115] Carsten Saathoff and Ansgar Scherp. "Unlocking the Semantics of Multimedia Presentations in the Web with the Multimedia Metadata Ontology". In: *Proceedings of the 19th International Conference on World Wide Web*. WWW '10. Raleigh, North Carolina, USA: ACM, 2010, pp. 831–840. ISBN: 978-1-60558-799-8. DOI: 10.1145/1772690.1772775.

[116] Alan Said and Alejandro Bellogín. "Comparative recommender system evaluation". In: *Proceedings of the 8th ACM Conference on Recommender Systems*. ACM Press, 2014, pp. 129–136. DOI: 10.1145/2645710.2645746.

[117] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. "Analysis of recommendation algorithms for e-commerce". In: *Proceedings of the 2nd ACM Conference on Electronic Commerce*. ACM Press, 2000, pp. 158–167. DOI: 10.1145/352871.352887.

[118] Claude Elwood Shannon. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[119] Gamgarn Somprasertsri and Pattarachai Lalitrojwong. "Extracting product features and opinions from product reviews using dependency analysis". In: *Seventh International Conference on Fuzzy Systems and Knowledge Discovery*. FSKD '10. IEEE, 2010, pp. 2358–2362. DOI: 10.1109/FSKD.2010.5569865.

[120] Eleni Stai, Stella Kafetzoglou, Eirini Eleni Tsiropoulou, and Symeon Papavassiliou. "A holistic approach for personalization, relevance feedback & recommendation in enriched multimedia content". In: *Multimedia Tools and Applications* 77.1 (2016), pp. 283–326. DOI: 10.1007/s11042-016-4209-1.

[121] Harald Steck. "Evaluation of Recommendations: Rating-prediction and Ranking". In: *Proceedings of the 7th ACM Conference on Recommender Systems*. RecSys '13. Hong Kong, China: ACM, 2013, pp. 213–220. ISBN: 978-1-4503-2409-0. DOI: 10.1145/2507157.2507160.

[122] Xiaoyuan Su and Taghi M. Khoshgoftaar. "A Survey of Collaborative Filtering Techniques". In: *Advances in Artificial Intelligence* (2009), pp. 1–19. DOI: 10.1155/2009/421425.

[123] Xin Su, Giancarlo Sperli, Vincenzo Moscato, Antonio Picariello, Christian Esposito, and Chang Choi. "An Edge Intelligence Empowered Recommender System Enabling Cultural Heritage Applications". In: *IEEE Transactions on Industrial Informatics* (2019), pp. 1–1. DOI: 10.1109/tii.2019.2908056.

[124] Ilya Sutskever, James Martens, and Geoffrey Hinton. "Generating text with recurrent neural networks". In: *Proceedings of the 28th International Conference on Machine Learning*. 2011, pp. 1017–1024.

[125] Charles Sutton and Andrew McCallum. "An Introduction to Conditional Random Fields". In: *Foundations and Trends in Machine Learning* 4.4 (2011), pp. 267–373. DOI: 10.1561/2200000013.

[126] Duyu Tang, Bing Qin, and Ting Liu. "Document Modeling with Gated Recurrent Neural Network for Sentiment Classification". In: *EMNLP*. 2015, pp. 1422–1432. DOI: 10.18653/v1/D15-1167.

[127] Karen H. L. Tso and Lars Schmidt-Thieme. "Empirical Analysis of Attribute-Aware Recommendation Algorithms with Variable Synthetic Data". In: *Studies in Classification, Data Analysis, and Knowledge Organization*. New York, NY, USA: Springer Publishing, 2006, pp. 271–278. DOI: 10.1007/3-540-34416-0_29.

[128] Andrew H. Turpin and William Hersh. "Why batch and user evaluations do not give the same results". In: *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, 2001, pp. 225–231. DOI: 10.1145/383952.383992.

[129] Iacopo Vagliano, Diego Monti, and Maurizio Morisio. "SemRevRec: A Recommender System based on User Reviews and Linked Data". In: *Proceedings of the Poster Track of the 11th ACM Conference on Recommender Systems*. Como, Italy: CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-1905/recsys2017_poster10.pdf.

[130] Iacopo Vagliano, Diego Monti, Ansgar Scherp, and Maurizio Morisio. "Content Recommendation Through Semantic Annotation of User Reviews and Linked Data". In: *Proceedings of the Knowledge Capture Conference*. K-CAP 2017. Austin, TX, USA: ACM, 2017, 32:1–32:4. DOI: 10.1145/3148011.3148035.

[131] Iacopo Vagliano, Cristhian Figueroa, Oscar Rodríguez Rocha, Marco Torchiano, Catherine Faron-Zucker, and Maurizio Morisio. "ReDyAl: A Dynamic Recommendation Algorithm based on Linked Data". In: *New Trends in Content-Based Recommender Systems*. CEUR Workshop Proceedings 1673. CEUR-WS.org, 2016, pp. 31–38. URL: http://ceur-ws.org/Vol-1673/paper6.pdf.

[132] Saúl Vargas and Pablo Castells. "Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems". In: *Proceedings of the Fifth ACM Conference on Recommender Systems*. RecSys '11. New York, NY, USA: ACM, 2011, pp. 109–116. DOI: 10.1145/2043932.2043955.

[133] Hongning Wang, Yue Lu, and Chengxiang Zhai. "Latent aspect rating analysis on review text data". In: *Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2010. DOI: 10.1145/1835804.1835903.

[134] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. "Learning Hierarchical Representation Model for Next Basket Recommendation". In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, 2015, pp. 403–412. DOI: 10.1145/2766462.2767694.

[135] Yiyu Yao. "Measuring retrieval effectiveness based on user preference of documents". In: *Journal of the American Society for Information Science* 46.2 (1995), pp. 133–145. DOI: 10.1002/(sici)1097-4571(199503)46:2<133::aid-asi6>3.0.co;2-z.

[136] Alexander Yates, James Joseph, Ana-Maria Popescu, Alexander D. Cohn, and Nick Sillick. "SHOPSMART: Product Recommendations Through Technical Specifications and User Reviews". In: *Proceedings of the 17th ACM Conference on Information and Knowledge Management*. CIKM '08. ACM, 2008, pp. 1501–1502. ISBN: 978-1-59593-991-3. DOI: 10.1145/1458082.1458355.

[137] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit Dhillon. "Scalable Coordinate Descent Approaches to Parallel Matrix Factorization for Recommender Systems". In: *2012 IEEE 12th International Conference on Data Mining*. New York, NY, USA: IEEE, 2012, pp. 765–774. DOI: 10.1109/icdm.2012.168.

[138] Mi Zhang and Neil Hurley. "Avoiding Monotony: Improving the Diversity of Recommendation Lists". In: *Proceedings of the 2008 ACM Conference on Recommender Systems*. RecSys '08. ACM, 2008, pp. 123–130. ISBN: 978-1-60558-093-7. DOI: 10.1145/1454008.1454030.

[139] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. "Deep Learning Based Recommender System". In: *ACM Computing Surveys* 52.1 (2019), pp. 1–38. DOI: 10.1145/3285029.

[140] Baoyao Zhou, Siu Cheung Hui, and Kuiyu Chang. "An intelligent recommender system using sequential Web access patterns". In: *IEEE Conference on Cybernetics and Intelligent Systems*. IEEE, 2004, pp. 393–398. DOI: 10.1109/iccis.2004.1460447.

[141] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. "Improving recommendation lists through topic diversification". In: *Proceedings of the 14th International Conference on World Wide Web*. ACM Press, 2005, pp. 22–32. DOI: 10.1145/1060745.1060754.

This Ph.D. thesis has been typeset by means of the TEX-system facilities. The typesetting engine was pdfLATEX. The document class was `toptesi`, by Claudio Beccari, with option `tipotesi=scudo`. This class is available in every up-to-date and complete TEX-system installation.