NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

JONAS GOUVEIA DE RODRIGUES

Bachelor of Computer Science and Engineering

JOINT SESSION-ITEM ENCODING FOR SESSION-BASED RECOMMENDATION: A METRIC-LEARNING APPROACH WITH TEMPORAL SMOOTHING

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
March, 2022

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

JOINT SESSION-ITEM ENCODING FOR SESSION-BASED RECOMMENDATION: A METRIC-LEARNING APPROACH WITH TEMPORAL SMOOTHING

JONAS GOUVEIA DE RODRIGUES

Bachelor of Computer Science and Engineering

Adviser: João Magalhães
Associate Professor, NOVA University of Lisbon

Co-adviser: David Semedo
Invited Auxiliar Researcher, NOVA University of Lisbon

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
March, 2022

Joint Session-Item Encoding for Session-Based Recommendation: A Metric-Learning Approach with Temporal Smoothing

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.7.1) [1].

Acknowledgements

This work is possible due to the contributions and guidance of Adviser João Magalhães and Co-adviser David Semedo, respectively Associate Professor and Assistant Researcher at NOVA University of Lisbon, the institution NOVA School of Science and Technology from NOVA University of Lisbon and finally the grant from the project "Converging broadcast and user generated content for interactive ultra-high definition services – COGNITUS" - H2020 ICT - with the grant agreement nº 687605.

"Whenever you find yourself on the side of the majority, it is time to reform (or pause and reflect)." (Mark Twain)

Abstract

In recommendation systems, a system is in charge of providing relevant recommendations towards users with either a clear target in mind or a mere vague mental representation. Session-based recommendation targets a specific scenario in recommendation systems, where users are anonymous. Thus the recommendation system must work under more challenging conditions, having only the current session to extract any user preferences to provide recommendations.

This setting requires a model capable of understanding and relating different interactions across different sessions involving different items. This dissertation reflects such relationships on a commonly learned space for sessions and items. Such space is built using metric-learning, which can capture such relationships and build such space, where the distances between the elements (session and item embeddings) reflect how they relate to each other. We then use this learned space as the intermediary to provide relevant recommendations. This work continues and extends on top of other relevant work showing the potential of metric-learning addressed to the session-based recommendation field.

This dissertation proposes three significant contributions: (i) propose a novel joint session-item encoding model with temporal smoothing, with fewer parameters and the inclusion of temporal characteristics in learning (temporal proximity and temporal recency); (ii) enhanced recommendation performance surpassing other state-of-the-art metric-learning models for session-based recommendation; (iii) a thorough critical analysis, addressing and raising awareness to common problems in the field of session-based recommendation, discussing the reasons behind them and their impact on model performance.

Keywords: Session-Based Recommendation Systems, Metric-Learning, Content-based Learning, Collaborative-Filtering, Joint Encoding, Session, Item, Temporal Smoothing.

Resumo

Em sistemas de recomendação, um sistema fica encarregue de fornecer recomendações relevantes aos seus utilizadores que podem ter, ou uma ideia concreta daquilo que pretendem ou apenas uma vaga representação mental. Recomendação com base na sessão dirige-se principalmente a um cenário específico de sistemas de recomendação, onde os utilizadores são anónimos. Ou seja, estes sistemas têm de ser capazes de funcionar em condições mais desfavoráveis, tendo apenas a sessão atual disponível como input do utilizador para efetuar recomendações.

Este contexto requer um modelo capaz de perceber e relacionar diferentes interações ao longo de várias outras sessões envolvendo diferentes itens. Esta dissertação reflete tais interações por via de um espaço comum, que é aprendido, para representar sessões e itens. Este espaço é construído usando *metric-learning*, técnica que consegue capturar tais relações e construir o espaço em questão, no qual a distância entre os vários elementos (*embeddings* de sessões e itens) reflete como estes se relacionam entre si. Usamos este espaço, que foi aprendido, como intermediário no fornecimento de recomendações relevantes. Este trabalho continua e extende para além de outros trabalhos relevantes na área que mostraram o potencial de aplicar *metric-learning* para o domínio de recomendação com base na sessão.

Esta dissertação propõe as seguintes três principais e significativas contribuições: (i) propõe um novo modelo de codificação sessão-item conjunto com suavização temporal, com menos parâmetros e com a inclusão de características temporais no processo de aprendizagem (proximidade temporal e recência); (ii) um desempenho de recomendação melhorado que ultrapassa outros métodos do estado-da-arte que utilizam técnicas de *metric-learning* para sistemas de recomendação com base na sessão; (iii) uma análise cuidada, que foca e tenta destacar alguns erros comuns neste campo de sistemas de recomendação com base na sessão, discutindo as razões por detrás de tais erros e o seu impacto no desempenho dos modelos.

Palavras-chave: Sistemas de Recomendação com Base na Sessão, Metric-Learning, Aprendizagem com Base no Conteúdo, Collaborative-Filtering, Codificação Conjunta, Sessão, Item, Suavização Temporal.

Contents

List of Figures

List of Tables

Glossary

phantom progress False claimed progress to the field in question. A problematic and common issue in the field of session-based and session-aware recommender systems. Such claimed progress cannot be verified or reproduced by other works. Usually it is due to methodological issues, possibly due to the competitiveness of the field. It is also present in papers published in highly-reputed conferences[2]. 5, 12, 13, 47, 74

Wizard-of-Oz For testing purposes of a new unimplemented technology, the user is generally given the impression to be interacting with a supposed theoretical intelligent computer application, where, instead he is interacting with another human simulating that same system. This evaluation method was proposed by [3]. 16

Acronyms

Bi-LSTM Bi-directional Recurrent Neural Networks of type Long-short term memory 11

CNN Convolutional Neural Network 7
CTR Click-Trough-Rate 11

MF Matrix Factorization 13
ML Machine Learning 7

NLP Natural Language Processing 8, 18

RNN Recurrent Neural Network 7, 8, 13
RR Retail Rocket dataset 13
RS Recommendation Systems 1, 2, 6, 10, 14
RSC15 RecSys 2015 Challenge dataset 13

1 Introduction

1.1 Context & Motivation

Conversational Systems [4–6] have attracted much research due to their potential in real use-cases. These conversational systems are a specific variant of Recommendation Systems (RS), to help users find the items they want through personalized item suggestions. This process involves two stakeholders, of which one is the user seeking a recommendation, and the other is the agent in charge of making such recommendations. Another similar scenario is an anonymous user actively browsing an e-commerce website (e.g., Amazon). The automated agent model utilizes the current session's past previous interactions after each event (e.g., click) to provide the following recommendation step.

In general, for recommendation systems, the user has a particular target item in mind, where sometimes this can be a specific item, or most usually the user is not exactly sure of what he wants, holding only a vague mental representation of it. The agent is in charge of making recommendations with the precise end goal of reaching the user's target item with the fewest recommendation iterations possible to maintain a good user experience. Therefore, the agent must possess enough intelligence to achieve this.

More recently, these systems have been used in more challenging contexts with limited interactions, i.e., in a session-based recommendation context. In this setting, users are anonymous, and our input comes through interactions from the current session, in the form of clicks, likes, or purchases in an online e-commerce fashion website social network domain like in Farfetch or previous sentences in a conversational system. Here, there is no notion of users, but instead, each session is considered independent, and in this world, it is as if each session defined a different user.

A session is composed of a sequence of events and can consist of a conversation or a shopping session in an online retail website retained with browser session cookies. The sessions' events can be exchanged messages or operations like a click, add to cart, or likes on items in a retail website.

This dissertation aims at the development of a Recommendation System for real-world scenarios like Farfetch's fashion e-commerce platform, similar to what is shown in

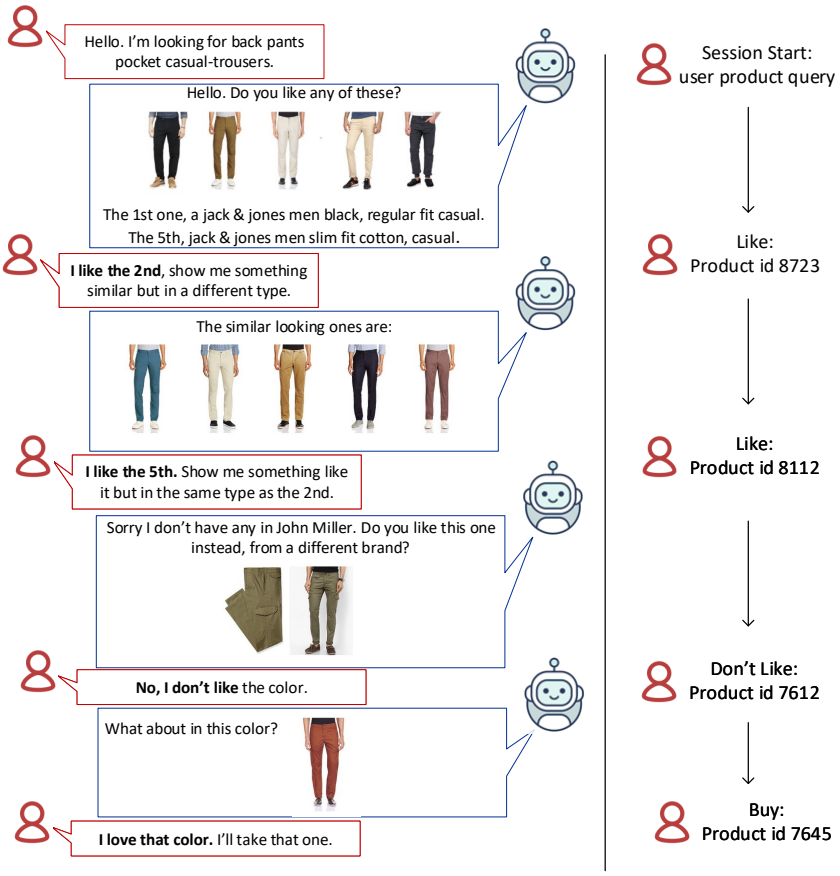


Figure 1.1: From Product Dialog Recommendation to Session-based Recommendation setting.

figure 1.1, targeted at scenarios where users are anonymous. In these specific contexts, we have an anonymous user seeking help towards finding a product and, on the other side, an automated agent whose job is to fulfill the user's needs, i.e., make recommendations until the user finds what he is looking for or the session ends. Despite these domains, the current work is not restricted only to the fashion domain of conversational systems. Instead, we opted for a more theoretical approach targeting the more broad field of metric-learning for session-based recommendation.

Metric-learning has proven to be an essential tool in the RS field, even for session-based RS [7]. Metric-learning allows us to learn a joint representational space of the users' sessions and items to recommend. With the current session projected on that learned space, we need to find the top-k closest items projected on that same space as the following recommendation (fig. 1.2). There is a relative lack of other research works that use metric-learning for session-based recommendation, and we believe that more attention should be given to this type of approach. Hence, this dissertation aims to address the session-based recommendation problem using metric-learning.

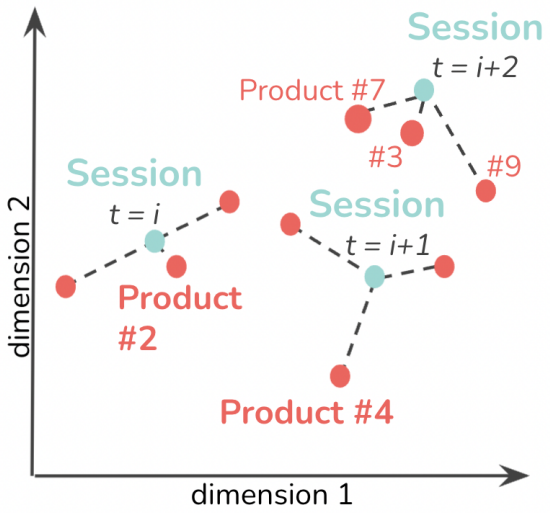


Figure 1.2: Example of live recommendation as the current session progresses, where in this case we are recommending the top 3 products that lie closer to the session embedding at the current step.

1.2 Problem Definition

In a session-based recommendation setting, an agent recommends items to a user seeking those recommendations. A session comprises a chain of ordered interactions made by anonymous users, where each interaction can also be described as an event. Each event regards an interaction involving one item, and this same event/interaction can be further described by its type. As we focus on just the recommendation system, we will handle structured input, as we can see on the right side of figure 1.1, where such system is being used as part of a conversational system. At time t of the recommendation process, we will recommend k products, ordered by their predicted relevance, using as our only input all the past events until *t-1* from the oldest to the most recent event that comprises our session. The agent uses this predicted recommendation list for its next recommendation step.

Understanding user behaviors becomes critical, more so in challenging scenarios where data is more limited, like in the case of session-based recommendation, where our input is only comprised of the previous interactions in the same session (e.g., clicks, likes, or applying for a job in a social network domain like LinkedIn), where the use of Recommendation systems becomes particularly desirable. A recommendation session is composed of a sequence of events, where each event is an interaction such as a *click*, an *add to cart*, or *like an item* at a retail website. In the case of streaming, such interactions can be a *view* or a *like* related to audiovisual content.

1.3 Challenges and Objective

This problem presents some exciting challenges like keeping track of the current session's state, producing quality recommendations, relating the session's state to the items, and which features play a decisive role in this process and how to extract them.

In order to address such challenges, the hypothesis is that the system should learn a session-item common space, i.e., where both items and sessions are represented in a commonly shared space through embeddings and metric learning. The current session is then projected on this learned space, and the recommendation comes from the items whose embeddings are closer to this session embedding. In this setting, our main goal is to:

> *Inspired by metric-learning literature, this dissertation's objective is to tackle session-based recommendation with joint session-item encoder architectures to compute the common embedding space. (fig. 1.2)*

This work aims to address the more broad spectrum of session-based recommendation, including but not limited to conversational recommendation systems, the fashion domain, or the streaming domain.

1.4 Contributions

Session-based recommendation using metric-learning explores the idea of learning a common space representing both sessions and items to be used for recommendation. However, the use of metric-learning to tackle this problem is relatively unexplored. Recently Twardowski, Zawistowski, and Zaborowski [7] proposed a metric-learning approach for this problem and have obtained promising results, so we further extend research in this direction, having accomplished the following contributions:

- **(a) Novel joint session-item encoding model with temporal smoothing.** We propose a novel architecture targeting item encoding, simpler than [7], where session and item encoders are jointly-learned in a common space, serving as an intermediary for session-based recommendation. We show that this contribution produces noteworthy improvements while at the same time significantly reducing the number of parameters involved. We also propose two different content-based learning techniques to kick-start the collaborative-filtering metric-learning model, although this did not contribute to enhanced performance. Finally, we propose to leverage the temporal characteristics in two dimensions: temporal proximity and temporal recency.

- **(b) Outperform other state-of-the-art metric-learning models for session-based recommendation.** We outperform other[7] state-of-the-art metric-learning models for session-based recommendation. We are also able to outperform other quite

robust session k nearest neighbors techniques like SKNN[8] and VSKNN[9] when taking advantage of the temporal features.

- **(c) Critical Analysis.** In this dissertation, we offer a critical analysis of the many dimensions of session-based recommendation in general. We analyze and bring awareness to certain topics we feel to be particularly important, like the computational complexity of the models as a relevant factor to consider besides the metric evaluation results, the in-depth analysis on the impact of dataset characteristics in the results, and a thorough discussion of the evaluation protocol for the recommendation domain. We also discuss several inconsistencies and mistakes commonly performed in this field related to the experimental protocol that creates significant confusion in the interpretation and comparison of similar research, a phenomenon called phantom progress[10, 11].

1.5 Document Structure

This document is organized as follow:

- In chapter 2 we present the study of the different concepts, techniques, and the selection of works sustaining the proposed work,
- In chapter 3 we formalize our problem and explain our contributions.
- In chapter 4 we conduct experiments to validate our contributions supplemented by a critical analysis of these same experiments results.
- In chapter 5 we present our conclusions and discuss future work directions.

2

Background and Related Work

An introduction and some background around other works in the field are needed to support this work. We first introduce the notion of neural networks in section 2.1.1, a state-of-the-art technique for machine learning and the method used for metric learning. Related to the same field, we also present attention mechanisms 2.1.1.1, a state-of-the-art technique for neural networks, as a possible direction to explore in our work as this technique has proven very useful in several learning contexts.

In order to build the representational space for recommendation, we use embeddings (sec. 2.1.3) to represent both the sessions and the products in a shared space.

In section 2.2, we will approach recommendation systems, first discussing some approaches for recommendation using collaborative-filtering (sec. 2.2.2), including techniques that use metric-learning.

Later in section 2.2.1, we will discuss the concept of *session* in recommendation systems, including the one we are focused on, the session-based recommendation domain, where sessions are anonymous.

Given our introduction of Recommendation Systems (RS), we then proceed to discuss conversational systems (sec. 2.3) in the context of RS. Even though we only focused this dissertation work on the RS, it is important to understand these systems that served as the initial motivation of this work and are an ever more desired variant of RS.

Finally, we go deep on our most important topic, metric learning (sec. 2.4), discussing: what it is; how it works; the different types of it through different loss functions; some possible improvements of it, mainly the use of adaptive margins (sec. 2.4.1) in the loss functions formulation; and how the use of different sampling techniques (sec. 2.4.2) can affect metric learning performance. We end with a discussion with several available and most used datasets 2.5 in this field of session-based recommendation systems, which is vital for our future work.

2.1 Background

2.1.1 Neural Networks

Neural networks are one of the most popular machine learning models in recent years. The original idea behind this conception was the attempt to simulate the human brain, first through the invention of the perceptron [12], followed by the imitation of a mesh of neurons, the multi-layer perceptron. The reason for its popularity is due to its ability to capture more complex relationships, and patterns previous classical and simpler models of Machine Learning (ML) could not handle well. Usually, they are advantageous for contexts involving a vast amount of data in unstructured form.

The complexity of Neural Networks also presents some tendency for overfitting, with the inability to generalize well when dealing with unseen data, as it learned patterns only representative of the data it was trained on. Some regularization methods were proposed to battle this overfitting tendency of these models, such as Dropout [13], L2 Regularization [14] and Batch Normalization [15]. These regularization techniques enabled the model to maintain its powerful ability to learn complex relationships in data while making it capable of generalizing well for unseen data.

Later on, some variants of it were proposed, like the Convolutional Neural Network (CNN) commonly used when dealing with images and Recurrent Neural Network (RNN) based architectures more adequate for problems with a sequential nature. In this work, Neural Networks, more precisely a CNN-based network with metric-learning is used to build a representational space where recommendation takes place.

2.1.1.1 Attention Mechanism

Bahdanau, Cho, and Bengio [16] contributed to one of the most important milestones for neural networks with the invention of the attention mechanism for neural networks. It was originally aimed at aiding neural networks in machine translation, where generating the next word during translation requires looking at the other words that compose the context, where some are more important than others in determining such context. To solve this, [16] proposed to mimic human attention in a neural network. Formally in a neural network setting, the attention mechanism maps a query and a set of key-value pairs to a vectorized output of weights, where the elements that the model should "focus"more have higher weights.

Attention was a revolutionary technique for neural network algorithms, making the models train faster and better results. The attention mechanism enables the model to focus on what is more relevant for the learning process.

RNN-based models are very limited in terms of parallelization due to their sequential nature, and learning from distant positions in a sequence has always been a problem for recurrent models. This was further improved with the insurgence of LSTMs[17], but the problem was never actually entirely solved. Vaswani et al. [18] proposed a simpler

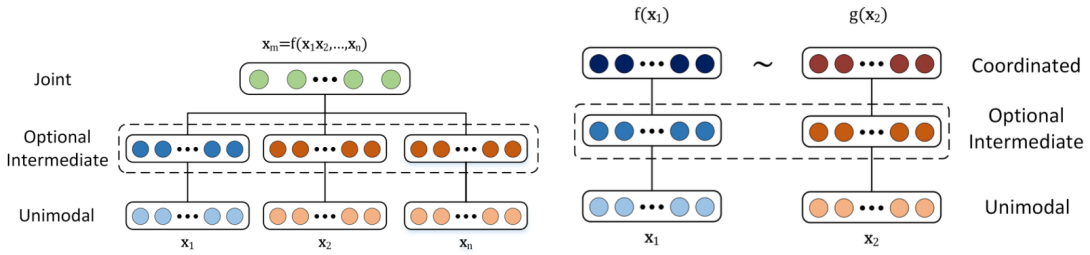
novel transduction model intended to replace dominant sequence transduction models, convolutional or recurrent neural networks with an encoder and decoder model. The proposed novel transduction model, the Transformer, is an attention-based architecture, where attention mechanisms become the basis of the whole architecture, more precisely, self-attention. The Transformer only requires only a single sequence pass across the model's components, instead of RNN-based models where the hidden state is rewritten at every step. The main advantage of this model is the self-attention mechanism, as it allows for the computations to be parallelized, and it is able to deal with long-range dependencies better than other RNN-based models. The authors concluded that this novel architecture based solely on attention, with multi-headed self-attention, can be trained significantly faster than other recurrent or convolutional methods, and they also achieved a new state-of-the-art. Later, other works based on the Transformer quickly became state-of-the-art like BERT [19] for NLP.

Attention techniques are adequate for our problem, as some events are more relevant than others in each recommendation step.

2.1.2 Multimodality

Modality refers to how something happens or is experienced, and a multimodal system contains and operates multiple modalities. Commonly it is compared to how we envision the world through taste, smell, vision, or touch. This multimodality feature allows conversational systems to become more complete and intuitive. In general, this approach obtains superior results compared to a single modality. Our aim in multimodal machine learning is to build models that can process and relate information from multiple modalities. We can easily find real-life examples that explain why multimodality is helpful. For example, when we listen to someone speak, we make use of the sounds they make when talking and from visual signals by observing the movement of the lips.

Baltrušaitis, Ahuja, and Morency [20] addresses this and divides this field into five challenges: representation, translation, alignment, fusion, and co-learning. Fusion has the challenge of joining information from different modalities to compute predictions and representation, which is aimed at learning how to represent multimodal data. We take advantage of the complementarity and redundancy of multiple modalities. A good representation is crucial for the performance of machine learning models. The similarity in the representation space should be according to the similarity between these concepts. It should deal well in obtaining this representation even in the absence of some modalities, provide the possibility to fill in missing modalities given the observed ones, and finally respect some properties like smoothness, temporal and spatial coherence sparsity, natural clustering. The authors proposed that multimodal representation consists of two forms, joint or coordinated, illustrated in figure 2.1. With joint representation, the unimodal signals are combined into the same representational space, where coordinated representation processes the signals separately but enforces certain similarity



(a) Joint Representation (b) Coordinated Representations

Figure 2.1: Structure of joint and coordinated multimodal representations from [20]

constraints, leading to a coordinated space. In other words, with joint representation, the same function processes all unimodal signals into the same representational space, while with coordinated representation, different functions process each unimodal signal into a coordinated space through cosine distance or maximizing correlation techniques. Joint representation is used mainly in scenarios where multimodal data is available during training and inference stages. The most straightforward method for this is just a concatenation of individual modality features, while others more complex rely on the popular neural networks.

2.1.3 Embeddings

An embedding is a mapping of an entity to a vector representation. Embeddings allow

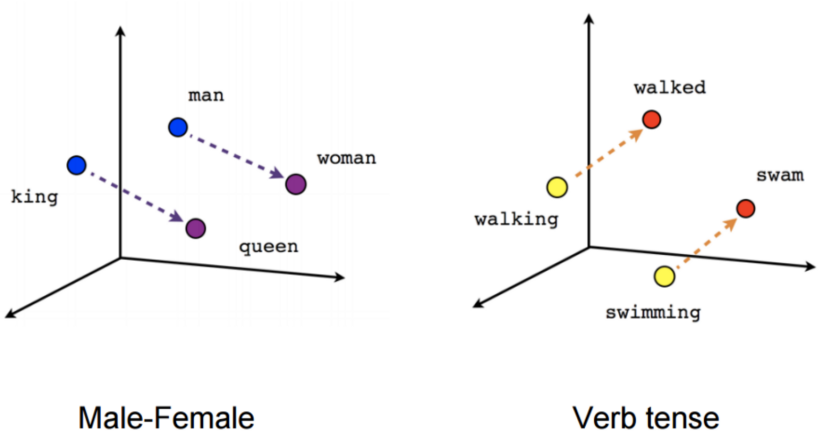


Figure 2.2: Word2Vec - Word Embeddings for NLP. [21]

us to represent discrete variables in a d-dimensional space (fig. 2.2). Their vector values, i.e., position on that space, are learned to take advantage of their relative representation in that space between each other. Usually, this happens in scenarios where we are mainly worried about the relative relationships between these entities (figs. 2.2 and 2.3).

Figure 2.3: We can easily see what each dimension might mean in these embeddings and the reason why these are useful in relative comparison scenarios [22].

Embeddings are the ideal representation for our case since we are interested in recommending the closest items through the relative distance between the current session and the universe of the products, and it can be accomplished through the projection of such embeddings.

Some good examples are the need to find similar movies or word embeddings for Natural Language Processing which try to represent the language semantics in that d-dimensional space like the popular Word2Vec[23]. A straightforward way to learn such embeddings is through Neural Networks.

Embeddings can have higher or lower dimensions. Higher-dimensional embeddings can capture more complex relationships but come at a high computational cost and the danger of leading to the curse of dimensionality machine learning problem or overfitting. With high-dimensional embeddings, when we try to compute the relationships between entities through a distance formula, usually euclidean distance, the distance between the points risks being almost equidistant; thus, machine learning models will have a hard time having any progress. On the other side, shallow dimensional embeddings might not be enough to represent more complex relationships in data. The choice of the embedding dimension to use is a crucial factor to consider, and it depends on several variables, like the domain where it will be applied or the data. Such embedding dimension parameter is usually set through grid search hyperparameter optimization or the more complex bayesian hyperparameter optimization.

2.2 Recommender Systems

In this section, we will approach the domain of Recommendation Systems (RS). Generally, an industrial Recommendation Systems has two steps: the candidate generation step and the candidate ranking [24] step.

The candidate generation phase requires lighter, more efficient, and simple algorithms

to generate candidates from the whole set. Conversely, the candidate ranking receives this smaller set of items from the candidate generation phase and applies more complex algorithms to rank it. This procedure provides a good balance between the computational burden and the quality of the recommendation.

Feng et al. [24] aimed at the candidate ranking task for Click-Trough-Rate (CTR) prediction. They verified that user behavior intra-session is highly homogeneous while heterogeneous in inter-session user behavior and proposed a session-based model for CTR prediction, Deep Session Interest Network (DSIN), which they claim capable of modelling the user's multiple session for this task. This model first divides users' sequential behaviors into sessions and then uses a self-attention network with bias encoding to model each session, which allows for a more accurate interest representation. This is followed by Bi-LSTM, in an attempt to address the sequential relation of contextual session interests.

Wu et al. [25] took advantage of the Transformer model and applied it in sequential recommendation taking in account the user preferences. To obtain better results than other state-of-the-art models, like SASRec[26] and Gru4Rec[27], they needed to add the Stochastic Shared Embeddings (SSE) [28] which stochastically replaces an embedding with another embedding with some probability during SGD, which ends up regularizing the embedding layers.

Hsiao and Grauman [29] addressed the issue of recommending a sub-optimal wardrobe to a user given a set of garments and accessories as a subset selection problem. To generate this they attempt an efficient optimization for a complex combinatorial mix-and-match outfit selection. This approach is able to achieve a reasonable result in a matter of seconds, providing the minimal set of items that provide the maximal result, i.e. help the customer get more for less. With maximal result they deconstruct it as three important factors, the compatibility between the items, the versatility (different styles) and also weight each style according to the user's style. The visual compatibility can be done in an unsupervised approach through a generative model trained on images of people wearing outfits "in the wild". The results obtained concluded that the system was able to mimic fashionistas, has potential for recommender systems, specially for cold-start recommendation. Dong et al. [30] also addressed the capsule wardrobe problem, focusing on personalization involving user preferences, body shapes and consumption habits. Their model has two main components, the user modelling component and the garment modelling component. The former handles user-garment compatibility, addressing personalization and the latter learns the garment-garment compatibility. As explained in 1, our work will contribute as the recommendation backbone for a conversational recommendation, for the fashion domain.

Latifi, Mauro, and Jannach [2] dissected the field of session-aware recommendation. Their work found that simpler nearest neighbors alternatives consistently outperformed recent neural network techniques, and session-aware models with access to long-term

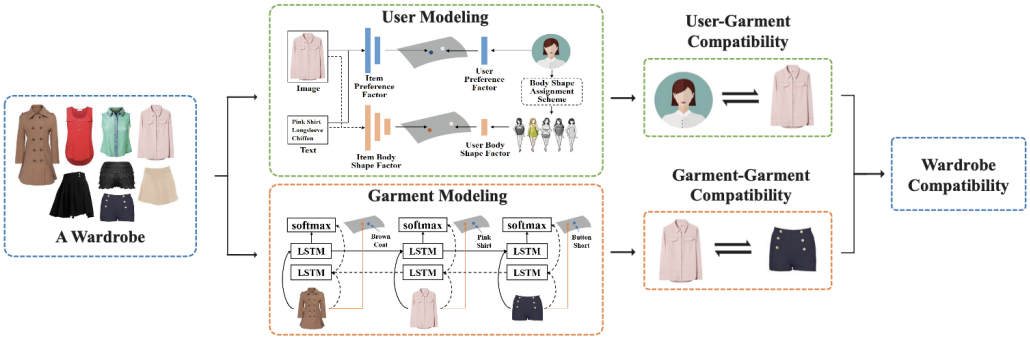


Figure 2.4: The architecture of the model proposed by [30]

data mostly did not surpass session-based models. They attribute this to potential methodological issues. Initially, the research community focused on rating prediction for recommender systems given a user-item rating matrix. They disregarded any temporal characteristics, either the order or time of such ratings. This trend has shifted, and research is now more focused on sequence-aware recommender systems. Here, instead of using rating prediction, we only use implicit user feedback (e.g., liking, viewing, or purchasing an item), i.e., time-ordered logs of recorded user interactions.

In their work[2], they researched these sequence-aware recommender systems that can be split into three main categories of recommender systems with the same prediction goal of the following user action: (i) sequential recommender systems that rely on user-item rating matrix as input; (ii) session-based recommender systems that have time-ordered logs of user interactions as input grouped in sessions where users are anonymous; (iii) session-aware recommender systems, where each session comes from a user's history actions (not anonymous). This work also revealed the potential of using *reminding* techniques, favoring the prediction of items that already happened in the session before, which led to promising results. The evaluation protocol also proposes executing a sort of data augmentation for the test set. For each sample session, they iteratively reveal each item as added input and proceed with a new evaluation step like in a real scenario. They also evaluate several recommender systems in two ways: (i) considering only the next item as ground-truth; (ii) considering all following items as relevant items, a more realistic scenario. Their work also criticizes other works in this field because they could not reproduce such claims and attribute this issue of phantom progress[10, 11] to methodological issues. They even noticed this for works published in highly-reputed conferences.

2.2.1 Session in Recommender Systems

This section explores how the session determines the type of recommendation system. As previously shown, such systems can be either session-based or session-aware.

Session-based Recommender Systems In a session-based setting, users are anonymous, and the goal is to predict the next user action given only the current session. The input is time-ordered logs of the recorded user interactions grouped in sessions. Thorough research in this field was ignited recently due to the release of open-source datasets like the RecSys 2015 Challenge dataset (RSC15) and Retail Rocket dataset (RR). GRU4REC [27] is a good example of this. This dissertation primarily addresses this variant of systems.

Session-aware Recommender Systems In a session-aware context, the users are not anonymous, as we know the previous user sessions when predicting their following action. This can be referred to as a personalized session-based recommender system, and there has been a lot of proposed models for this context, primarily based on the RNN architecture.

Latifi, Mauro, and Jannach [2] studied state-of-the-art models of sequential, session-based, and session-aware models and concluded that current state-of-the-art session-aware recommender systems provide the worst results. They are ineffective at using the long-term preference information in data, and actually, methods based on nearest-neighbors for session-based recommendation provided the best results, even when compared against deep learning methods. Although authors of several papers that proposed different session-aware recommender systems claimed progress, this is likely due to methodology errors resulting in phantom progress[10, 11], a common issue in the research community of this field. To tackle this, the authors of [2] also proposed an open-source framework for the evaluation of these session-aware recommender systems.

2.2.2 Collaborative Filtering

Collaborative filtering is a technique that aims to provide personalized recommendations based on filtered information of user data and their interactions with items. To illustrate, if we assume a scenario where users that usually buy items A and B also buy item C, a collaborative filtering model filters and finds these patterns, which would, in this case, help recommend item C to those that bought items A and B. This is used, for example, on Netflix, which even has an open contest on this matter called Netflix Prize[1].

One of the main tools for collaborative-filtering are Matrix Factorization (MF) techniques, which are not to be mistaken with metric-learning since the dot product involved in matrix factorization does not satisfy the triangle inequality. When this constraint is not respected, the distance does not represent dissimilarity as it should [31]. Matrix factorization can predict the ratings but cannot rigorously determine the relationship between user-user and item-item.

Instead of using matrix factorization, Hsieh et al. [32] addressed a collaborative-filtering setting but now applying metric learning. Using similarity propagation, they

[1] https://www.netflixprize.com

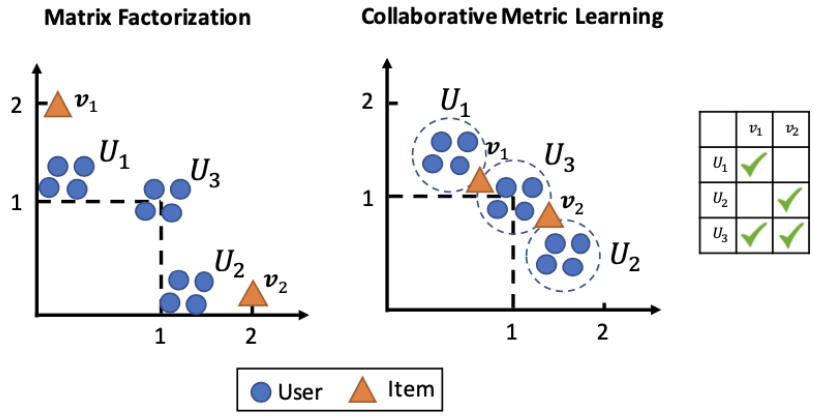


Figure 2.5: Collaborative Metric Filtering is able to better represent the space with the user-user and item-item relationships that MF does not consider.

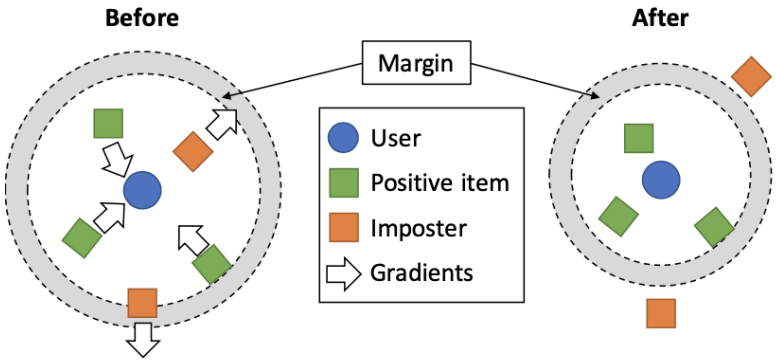


Figure 2.6: Impostors are pulled away from the perimeter.

learned a joint metric space, encoding users' preferences and user-user and item-item similarity, propagating the known similarity information to the unknown relationship pairs. Besides the advantage of metric learning respecting the triangle inequality while matrix factorization does not, the authors proved that taking advantage of the item features, i.e., content-based learning, also led to better results.

2.3 Conversational Recommender Systems (CRS)

Conversational systems usually involve two stakeholders, the agent and the user, which interact through dialogues in a conversational setting. These systems are a form of Recommendation Systems (RS), where the agent tries to understand the user's preferences and, through a conversation, in natural language, recommend products to the user. The user is also expected to state his preferences in this conversation which could either be voluntary or when explicitly asked by the agent. A session could be interpreted as the

utterances of the conversation.

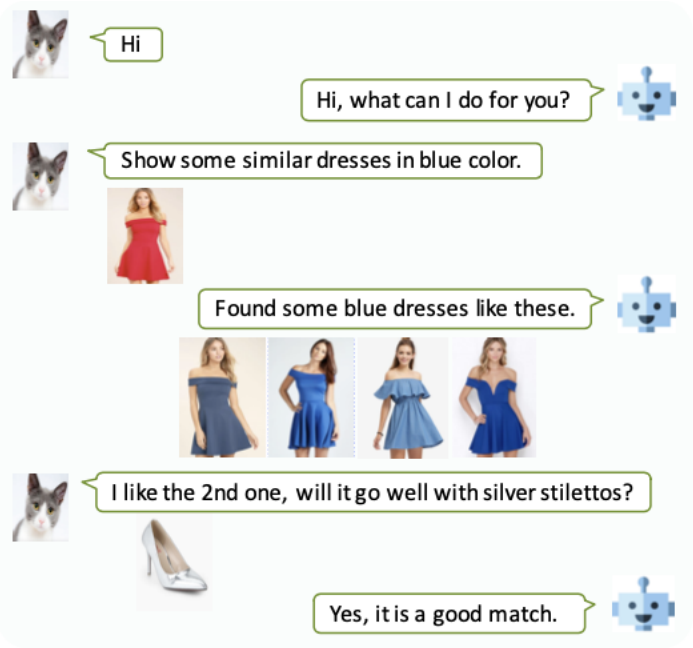


Figure 2.7: Example of a Conversational System for Recommendation from [5]

The following models were proposed to tackle the task of recommendation in a conversational setting in the fashion domain:

2.3.1 End-to-End CRS

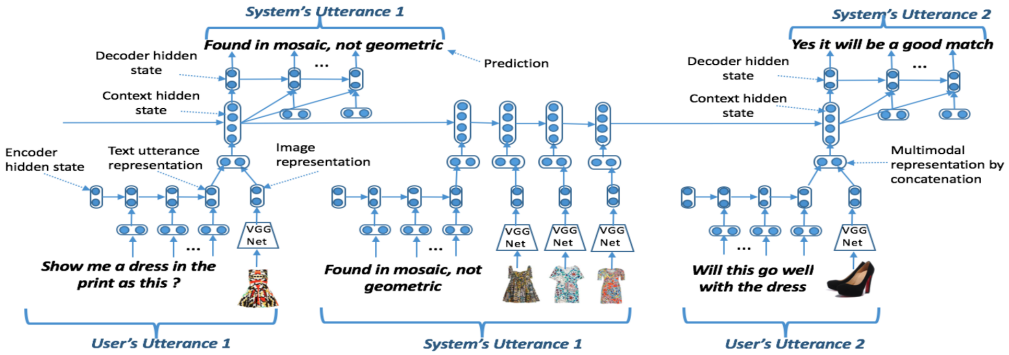


Figure 2.8: MHRED model for text prediction task from [4]

MHRED Saha, Khapra, and Sankaranarayanan [4] in order to build the MMD dataset, used a novel Multimodal Hierarchical Encoder Decoder (HRED) with an encode-attend-decode paradigm (fig. 2.8). This model is a multimodal neural network able to incorporate, at the same time, as input, both text and image content. The use of both modalities,

text and image, besides more closely mimicking real-life conversational systems and providing a better user experience, also led to better results in the isolated image or text tasks. They tried using attention in the model, but unexpectedly it led to worse results.

KMD Liao et al. [5] proposed the Knowledge-aware Multimodal Dialogue model, which represents the products in a continuous space (embeddings). It employs attention on the knowledge base maintained on memory networks and is based on the HRED model architecture. They used a taxonomy-based visual semantic model to explicitly represent fashion concepts using an EI Tree structure. This model also uses Reinforcement Learning and provides better BLEU scores than previous baseline models.

UMD The Hierarchical User Attention-guided Multimodal Dialog System [6] uses a Bi-directional LSTM with attention applied to the text in order to select the product's attributes the user is specifying. The authors propose a novel hierarchical encoder to learn the taxonomy-guided attribute-level representation of product images using a CNN, focusing on the user requirements, more specifically, the product's attributes requirements.

ReDial Originally targeted at the movie recommendation domain from the paper that released the ReDial dataset [33], the authors also built a modular model which is based on the HRED architecture. Their model uses a switching decoder to output the next token as a word or a movie and, after each dialogue act, detect if a movie entity has the seeker's sentiment regarding that entity, allowing for sentiment analysis.

Although we assume, as given, the input session representation, we could take advantage of this model and take the context hidden state from the inner RNN layer on figure 2.8, which would represent our session, project this on our learned d-dimensional space and obtain the closest items as our next recommendation prediction. This could enable our work to perform in a fully conversational system. Nevertheless, more powerful systems based on self-attention, as we have discussed in section 2.1.1.1 might be a better solution.

2.3.2 Eliciting User Preferences

A coached conversational setting allows the users to give more rich feedback and preference elicitation for conversational systems in a natural yet structured form.

Radlinski et al. [34] addressed this issue and collected a dataset with over 12,000 annotated utterances between a user and an assistant discussing movie preferences in natural language using a Wizard-of-Oz[3] methodology, focusing on preference elicitation instead of task completion, to collect natural yet structured conversational preferences. They proposed a new robust approach for eliciting preferences but in a natural form, which avoids mirroring uncertain assumptions of how users describe their preferences realistically, and they also studied how people naturally express preferences in a conversational setting.

Kovashka, Parikh, and Grauman [35] also studied the use of feedback by the user proposing a coached version, Active Whittlesearch, that tries to guess which question the system should ask the user in terms of a specific attribute of an item. To accomplish this, they use a probabilistic and information gain model that tries to guess the user's answer and obtain the attribute and item that would reduce the system's uncertainty/entropy the most. In other words, they try to predict the following question to ask that would get the system closer at guessing the user's target item. This kind of feedback-seeking mechanism is called relative attribute feedback.

Habib, Zhang, and Balog [36] also addressed this field, but they also tried to account for the possibility of changing user preferences by using dialogue intents.

Binary Relevance Feedback In this type of feedback, the user states if the presented recommendation is relevant or irrelevant. This approach is severely limited since it does not help the system learn exactly what is relevant in the stated product. Eventually, the model will learn, but it requires more samples and iterations. This presents an advantage in some specific cases, like at the end when the reference images being presented are very close to each other, and it may be more appropriate and efficient to use binary feedback or if the system is far from reaching the target image.

Relative Attribute Feedback In this setting, the user gives more fine-grained feedback, giving relative feedback of the attributes from the recommended items. For example, this can lead to the user requesting a shinier product than one of the images presented but darker than another. This technique contributes more to model learning since we specify what we want in terms of the product's attributes. Usually, relative feedback presents better results than binary feedback, except for cases where we are very close or very far from the target item.

Hybrid Feedback This approach combines the other two feedback techniques and is usually the best approach[35].

2.4 Metric Learning for Session-based Recommender systems

A common technique resides in the use of metric learning to build the embedding space. Metric learning attempts to map data, which in our case will be items and sessions, to a common embedding space, in which similar elements are close together, and dissimilar elements are far apart. Several loss functions allow models to accomplish this. The first released and currently the most known ones are the contrastive loss[37], and an extension of it, the triplet-loss[38] function.

The contrastive loss is the classical pair-based method and the first metric learning loss function proposed in 2006 by [37]. This loss function tries to make the distance between negative pairs larger than a predefined threshold while also making the distance

between the positive pairs smaller than another predefined threshold, hence being pair-based. These threshold values lead to lower variance, resulting in less expressiveness on the final embedding space.

The triplet-loss function, also released in the same year [38], tries to fix this issue. It uses 3 elements, an anchor element and a positive and a negative element, w.r.t. the anchor. The goal of triplet-loss is to minimize the distance between the anchor and the positive samples while maximizing the distance between the anchor and the negative samples. It is based on the hinge-loss and usually requires a fixed margin in its formulation. The triplet-loss function intends to make the distance between the anchor and the positive sample lower than the distance between the anchor and the negative sample by a predefined margin. This is an improvement of the contrastive loss[37] function, which assumes a predefined threshold to separate similar and dissimilar samples pairwise. Triplet-loss, instead, has the flexibility to distort the space, tolerating outliers and adapting to different levels of intra-class variance for different classes.

Contrastive loss[37] also enforces all positive samples to be as close together as possible, where instead triplet-loss is more relaxed, only requiring positives to be closer than negative samples. This more relaxed approach suffices on domains where we are only worried about relative relationships.

Following these two metric learning embedding losses, other ones have been proposed, always claiming to surpass the previously proposed state-of-the-art loss functions, but as verified by [39] recently, this is not the case. This is due to unfair comparisons, wrong accuracy metrics, training with test set feedback, or even more concerning, adjusting the hyperparameters to the test set results, skipping the use of validation sets, breaching one of the most basic machine-learning principles. The authors found that these released papers provide a slight improvement relative to the original contrastive and triplet loss functions, and to aid in future work, they also provided some guidelines and a framework to evaluate future research work in this field correctly.

Twardowski, Zawistowski, and Zaborowski [7] also addressed metric learning, in their case for session-based recommender systems, a type of system that is usually tied to time and sequence models, with a sequential nature similar to Natural Language Processing (NLP), where the predictions are based solely on the user's actions. They consider a session as a sequence of events/interactions with items, which leads to sessions being comprised of sequences of item identifiers in their collaborative setting. Their proposed method learns a distance function to accurately represent the dissimilarity between the sequence of users' events and the next interacted item, which allows a common space where sessions and items are represented. During inference, a session is projected on this learned space, and the top-k closest items in that same space will comprise the following ranked recommendation of the system.

Their proposed model comprised two separate encoders, the session encoder, and the item encoder, both projecting to the same common space. Specifically for the session encoder, they tried different types of neural networks RNN and CNN based, but the

results did not conclude a clear winner across the different datasets they used. For metric-learning, they used the robust triplet-loss[38] function and the more recent Smoothed Deep Metric Learning (SDML) Loss[40] function better suited for more noisy scenarios. For the triplet-loss function, they tried two different variants, one where temporal proximity was embedded in the loss-function, which they referred to as weighted triplet-loss, and another variant applied a swapping technique for the anchor. During training, the swapping technique replaces the anchor with a positive sample, accounting for the distances between the positives and negatives, similar to the contrastive-loss[37].

When sampling, their approach[7] was to split the session randomly, with the first part being used as input and the following actions with items used as positive samples. Conversely, negative items are randomly sampled from the dataset.

Their[7] ablation study concluded that the weighted triplet-loss led to superior results, which was not the case for the swapping technique. Besides this finding, they also found that using a single shared item embedding layer used by both session and item encoders was superior to having a separate item embedding layer for each encoder.

They[7] observed that their method gives a broader spectrum of recommended items than other simpler frequency-based or nearest neighbors baselines and concluded that metric learning portrays as an essential ingredient for session-recommender systems.

2.4.1 Adaptive Margins

The original loss functions for metric learning required the choice of a predefined margin value before training, further limiting the learning process. As shown by [41], previous approaches based on the hinge loss could not adapt while training as the subspace structure changed, and some works also enforce the need for a minimum fixed margin between different categories in the cross-modal subspace, which is not the ideal representation since similarities between different categories are different pairwise.

This issue was recently addressed by [41] which formulates a triplet-loss with an adaptive maximum margin superseding the original formulation in expressiveness and results.

> "The rationale is to solve the heterogeneity problem by learning a common space in which semantically equivalent instances will be structured close together."(Semedo and Magalhaes [41])

The authors[41] proposed a novel adaptive neural structuring subspace learning model (SAM) when addressing the task of cross-modal retrieval. Instead of relying on a fixed margin that separates categories on the subspace, their approach uses a scheduled adaptive margin function, smoothly activated. The model adaptively infers this margin, reflecting triplet-specific semantic correlations, considering the incremental learning behavior of the neural network, and enforcing category cluster formation. This organizes instances by adaptively enforcing inter-category and inter-modality and allows for better

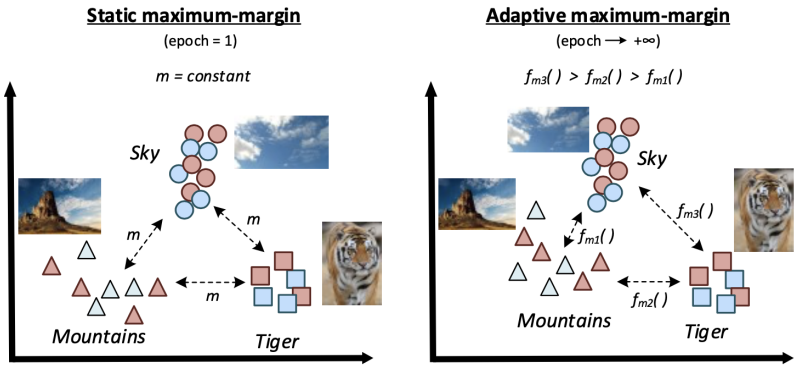


Figure 2.9: Comparison between the static margin versus the adaptive margin and the implications on the resulting learned subspace by [41]

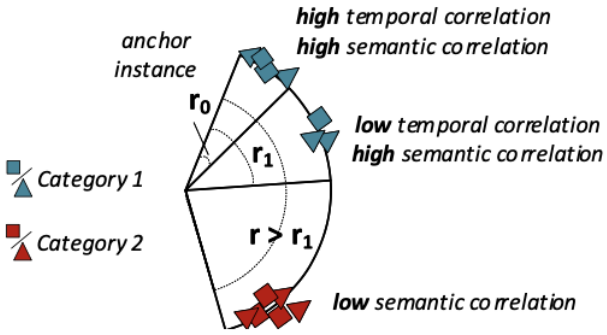


Figure 2.10: The incorporation of time and its impact on the dissimilarity computation between entities by [42].

quality embeddings (fig. 2.9) since it considers the embeddings' semantic similarities on both modalities correlations and does not neglect embeddings semantic similarities as with the fixed margin approach. The result, as intended, was a new method to effectively obtain a semantic subspace organization with better results than other state-of-the-art methods. We can consider this an improved triplet-loss formulation.

Later released by the same authors, [42] proposed a further improvement on top of the authors' previous work[41] that used an adaptive margin function with the addition of temporal traits into the cross-modal embedding space. The rationale stands behind the idea that information changes over time and the vital role that temporal traits have in finding how different modalities are correlated. To capture these temporal correlations, neural networks were used for cross-modal embedding learning to learn complex temporal correlations in data. The adaptive margin is used to quantify the temporal correlation between these instances, therefore not only is data organised by cross-modality correlations and categories but also from temporal correlations. To achieve this, the authors[42]

consider the existence of two dimensions, the semantic and temporal dimensions as illustrated in figure 2.10, resulting in two loss functions that are combined, in a differentiable loss function, and jointly optimized. After giving input in a modality, we get the top-k nearest-neighbors in the opposite modality to obtain the result. Two approaches were presented, both outpacing other state-of-the-art methods, where the choice between one or another depends if the modalities have only one or more modes. As in the paper:

> The ability to incorporate temporal traits into this cross-modal embedding space might indicate that, in our case, the product's attributes could be modelled in a similar manner. (Semedo and Magalhães [42])

2.4.2 Sampling

Sampling is an essential factor to consider when using metric-learning techniques.

Wu et al. [43] specifically addressed this topic of sampling for deep metric learning, claiming that the choice of the sampling technique is as important as the choice of the loss function. They proposed a new sampling technique, Distance Weighted Sampling (fig. 2.11), which corrects bias while controlling variance simultaneously. After doing

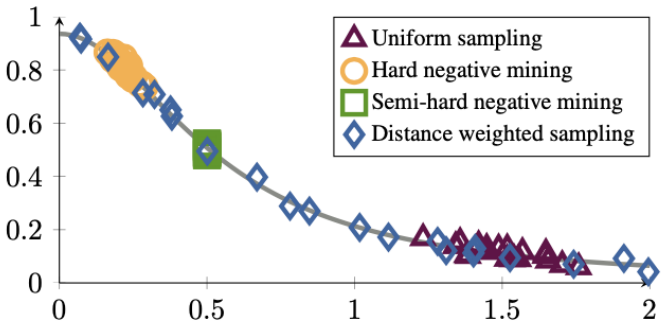


Figure 2.11: x-axis represents the distance between samples and y-axis represents the computed gradient. We can notice how the Distance Weighted Sampling technique does indeed maintain a stable gradient variance while at the same time selecting samples without being distance biased like the other methods, image by [43].

some benchmarks, the authors claimed to obtain similar results when applying the same sampling technique to either contrastive loss or triplet-loss. Therefore, contrary to what was believed, triplet-loss is not just better than contrastive loss because of the loss function itself but also because of the sampling methods associated with it. This work proves that improvement can also arise due to the choice of the sampling technique. Thus other works in the field should consider this choice to be as important as the choice of the loss function.

Twardowski, Zawistowski, and Zaborowski [7] also experimented with different sampling techniques and concluded that their positive-negative sampler led to better results, which is the same sampling technique used in this dissertation.

2.5 Datasets for Session-based Recommendation

ReDial REcommendation through DIALog [33] is the only real-world dataset, two party-conversational corpus in natural language available with over 10,000 conversations applied to movie recommendations. The elements involved are the recommendation seeker (agent) and the recommender (user) collected through Amazon Mechanical Turk (AMT) with a custom developed interface. To collect it, the movie seeker needs to explain what kind of movies he likes and ask for movie suggestions. The recommender must attempt to understand the seeker's movie tastes and recommend movies. Movies are explicitly tagged for later matching with DBpedia on semantic web in order to avoid errors if these where matched on a natural language form. The data is validated with feedback from both the agent and the user, at each movie, if the movie was mentioned by the seeker or suggested by the agent, if the user has seen the movie and finally if the user liked it. The authors claim the size of the dataset is likely not enough for end-to-end neural model training, which was addressed by them, with a modular version with fully neural architecture. Even though it does not have movie attributes, they are tagged and could be extracted using DBPedia on our end.

RR (Retail Rocket) is a dataset from an e-commerce company that contains user interactions of types "view", "addtocart"and "transaction". It has also information about the items' attributes in a timestamped form (e.g price as an attribute can fluctuate frequently) and their categories represented in a hierarchical form. This dataset is from the Retail Rocket company that has its business model focused in assisting retail operators in personalized real-time recommendations. This dataset was released in order to motivate researches in the field of recommender systems with implicit feedback. For confidential reasons most of the data is hashed.

XING is a dataset extracted from a job-posting social network that comes by that name released for the ACM RecSys Challenge 2016. It is a sort of LinkedIn social network, users use it to find jobs and recruiters to find workers. The goal of this dataset is the development of recommendation systems that would recommend relevant job posting to the user. Data is obfuscated for privacy reasons. It provides the attributes of both user and job posting items. The interactions can be of types, click, bookmark, reply and delete.

RSC15 (ACM RecSys Challenge 2015) is a dataset made available for the ACM RecSys Challenge 2015. The values were changed before open-release to account for privacy. It contains user interactions on items that can be clicks or purchases. It has the sessions already identified, contrary to the RR dataset where we only have the users' interactions and their timestamps, requiring extra work to estimate the sessions based on the timestamp gaps. Unfortunately the only item attributes on this dataset are the category

Table 2.1: Datasets descriptions

Name	Text	Image	Item Attributes	Domain
ReDial	✓	✗	✗ (But can be mined)	Movies
RR	✓	✗	✓	E-Commerce
Xing	✓	✗	✓	Job-listing social network
RSC15	✓	✗	(Only category & price)	E-Commerce

and price, not ideal for our end goal of content-based learning (taking in account item attributes).

2.6 Critical Summary

- The use of metric learning for session-based recommendation presents a big potential and quite an uncharted field in research. Twardowski, Zawistowski, and Zaborowski [7] approached this and obtained good results when applying metric learning for the session-based recommendation admitting that there is still a lot to explore, they only slightly touched this subject. Metric learning is ideal in situations where we are mainly interested in predicting something in which relative comparisons are enough. This is the case for recommendation, we can address recommendation by projecting the inputs on a space and through metric learning guarantee the correct construction of that space, where the relationships between these projections translate to the dissimilarities between them. Having this space, it then becomes quite intuitive and natural to perform recommendation by taking advantage of their relationships/distances, but this is still underrated in the research field for recommendation.

- Techniques used within Metric Learning that allow for better expressiveness of the projected space generally lead to better results. One of the most promising techniques are the use of adaptive margins that will lead to a more expressive model, thus generally a more robust model better able to represent other more complex features like time [41, 42]. Unfortunately a lot of the research that uses metric learning still do not make use of these adaptive margins. We can almost say, analogously, that adaptive margins is to metric learning what attention is for neural networks. Although there are works, like the very recent work by [31] that do take advantage of metric learning with adaptive margins for recommendation, they are still a minority in this research field.

- The fierce competition in the whole machine learning spectrum has been one of the reasons leading to breakthroughs in several branches of this field. However, this has also augmented pressure in researchers to excessively outgrow other state-of-the-art works, as the inability to do so might equal less citations and their work not being so

much of interest to the public research community. This creates a temptation to bias the results in order to conclude meaningful improvements that are actually only the result of incorrect methodology [39]. A common and one of the most serious examples of this, is the use of only train and test stages, skipping validation. This is wrong because since we chose a certain model as it gave the best results on the hyperparameter search, any results also evaluated on this model's test set after this prior selection is biased. This breaches one of the most basic commandments of Machine Learning and only leads to obstacles for other researchers. In our work we will not adhere to these bad practices.

3

Joint Session-Item metric-learning

This chapter starts by formalizing our problem and introducing the notation used in the remainder of this dissertation. Then, we formulate and discuss the distinct fundamental approaches for session-item subspace learning, where we propose the following contributions: (a) a joint-session item encoder, (b) the combination of collaborative-filtering with content-based learning in a hybrid learning approach, and finally, we propose (c) to add temporal re-ranking on top of the metric learning models, based on the concept of recency, where the most recent events have higher importance for the next prediction.

3.1 Problem Formalization

In a dataset with N_s sessions $\mathbf{S}$, each session $s_i \in \mathbf{S}$ corresponds to a sequence of N_{s_i} events e_t^i performed by an *anonymous user* as shown in the following equation 3.1:

$$\mathbf{S} = \{s_i\}_{i=1}^{N_s} \quad \text{where} \quad s_i = \left[e_t^i\right]_{t=1}^{N_{s_i}} \tag{3.1}$$

A session is composed of events that happen at different session steps, never coinciding because two events cannot happen simultaneously. Each session step is only related to a single event and vice-versa (e.g., the first event of a session defines the first session step; conversely, the first session step also defines its first event). Given this, we can formulate this intra-session order as the session time step t that starts counting from the beginning of the session.

Each event is a logged interaction from a user towards a particular item it_j. The type of interaction can describe the interaction itself, while we can also use item metadata to identify its intrinsic properties. An item $it_j \in \mathbf{I}$ can appear in multiple sessions and is uniquely identified in the dataset.

The formalization of events and items is dependent on the learning strategy chosen: collaborative-filtering only strategy or a strategy that combines both collaborative-filtering and content-based learning. We will discuss this in more detail along with the chapter.

In a collaborative-filtering only setting, an event is only defined by its session time step t, plus the unique ID of the item the event concerns, as we define below in equation 3.2:

$$e_t^i = it_j \quad \text{where} \quad it_j = ID_{it_j} \iff \iff e_t^i = ID_{it_j} \tag{3.2}$$

For the combination of collaborative-filtering with content-based learning, additional semantic event and item-related properties are considered. Events e_t^i are now characterized by its session time step t, the type of event/interaction involved (e.g. *click*, *view*, *buy*, etc.) and the item involved it_j, where the item representation is not restricted to its unique ID but is also described by a set $\mathbf{P}$ of N_p meta-data properties p_k,

$$e_t^i = \left[e_{t_{type}}^i, it_j\right] \quad \text{where} \quad it_j = \left[ID_{it_j}, \mathbf{P_j}\right] \tag{3.3}$$

where p_k may correspond to the *category of the item*, *color*, *price*, and others, depending on the dataset domain. A property p_k can be defined by its name (e.g., color) and value (e.g., blue),

$$\mathbf{P} = \{p_k\}_{k=1}^{N_p} \text{ where } p_k \in \{category, \ldots, color\} \tag{3.4}$$

For ease of reading, we summarize the introduced notation in table 3.1, with the corresponding description.

In session-based recommendation, given a session $s_i = \left[e_1^i, \ldots, e_t^i\right]$, at a certain time step t, the goal is to correctly predict the next item it_j to be recommended, where $it_j \in e_{t+1}^i$.

3.2 Session-Item Common Subspace

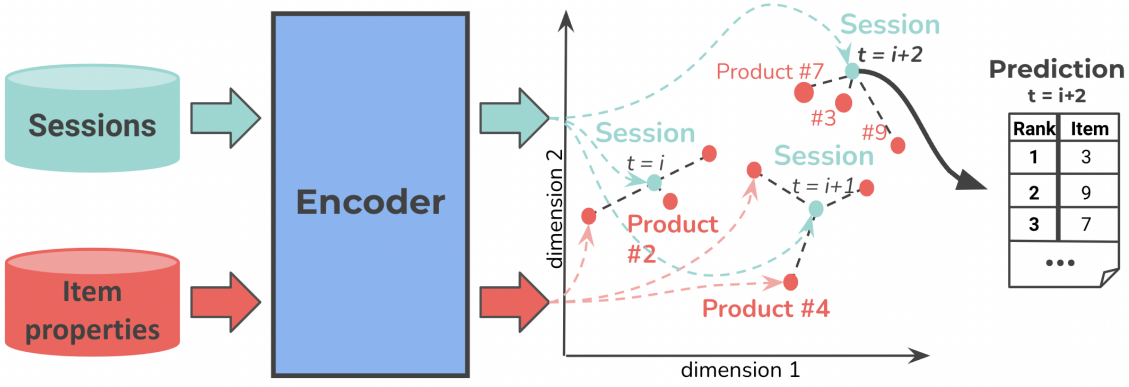


Figure 3.1: Diagram illustrating a general model using collaborative and content-based learning for session-based recommendation with metric learning.

We propose a deep metric-learning framework that produces such recommendations by navigating over a semantically enriched jointly-learned session-item space that embeds content semantic properties. We obtain a semantically enriched novel joint embedding

Variable		Description
Session	$\mathbf{S}$	Set of all sessions in the dataset
	s_i	A session sampled from $\mathbf{S}$
	N_s	Number of sessions in the dataset
Event	e^i_t	The t-th event of a session s_i
	N_{s_i}	Number of events in session s_i
	$e^i_{t_{type}}$	The type of the t-th event of a session s_i
Property	$\mathbf{P}$	Ordered set of the item properties
	N_p	Number of different item properties
	p^{key}_k	The k-th property key name
	p^{value}_k	The k-th property key value
Item	$\mathbf{I}$	The set of all Items in the dataset
	it_j	A sampled item from the dataset, from $\mathbf{I}$
	ID_{it_j}	The unique identifier number for item it_j
Encoding	z	A sequence of tokens comprising a single event (eq. 3.7)
	Z_i	A sequence of tokens that represent the i-th session (eq. 3.8)
	D	Dimension of the embeddings for sessions and items

Table 3.1: Dissertation notation description.

space with this framework where sessions and items lie. Figure 3.1 shows a diagram illustrating an abstract view of the proposed framework applied to session-based recommendation problem.

Recommendations are then made by navigating across concepts (Sessions $\mapsto$ Item) in the metric-learned common subspace. Specifically, as the session unfolds, its embedding will also evolve accordingly, such that new recommendations can be made by finding the closest item neighbors in the joint space.

The proposed session-item common space will be jointly learned by constraining session and item embeddings using metric-learning. In this setting, an embedding of a session is constrained to lie close to an item if that item is in the next events to follow. More concretely following Twardowski, Zawistowski, and Zaborowski [7], a certain session $s_i = \left[e^i_1,\ldots,e^i_t,\ldots,e^i_{N_{s_i}}\right]$ at a step t, will be constrained to lie close to it_j if $it_j \in [e^i_{t+1},\ldots,e^i_{max(N_{s_i},t+pos-1)}]$, where pos is the hyperparameter that sets the maximum amount of positive/relevant items to sample from the ground-truth (the next events). The sampling technique of positive and negative samples is further described in sec. 3.5.2. In section 3.5.1, we will also show two interesting scenarios where the items inside this

window can be equally relevant or alternatively have different relevance weights.

3.3 Session-Item Encoding

To learn such common session-item space, we define two neural encoder functions:

$$f_{session} : s_i \mapsto \mathbb{R}^D \quad , \qquad f_{item} : it_j \mapsto \mathbb{R}^D, \tag{3.5}$$

where D is the common embedding size, $f_{session}$ denotes the session encoder network and f_{item} the item encoder.

The $f_{session}$ neural encoder takes as input a session s_i and outputs a D-dimensional embedding.

The encoder f_{item} takes as input an item it_j and outputs a D-dimensional contextualized item embedding.

These two encoder functions are jointly optimized using metric-learning.

In this chapter, we will present and discuss two different encoding approaches for f_{item} and $f_{session}$, either using two cross-encoders or a single joint-encoder for both concepts, followed by a more deep exploration and discussion of each strategy.

3.3.1 Cross-Modal Encoding

In this section we explain one of possible encoding strategies for f_{item} and $f_{session}$. The cross-modal encoding was the encoding strategy proposed by Twardowski, Zawistowski, and Zaborowski [7]. In this encoding technique, there are two separate encoders, one for the items and another for sessions, respectively f_{item} and $f_{session}$.

The item encoder f_{item} is comprised of two main sections, an item embedding layer followed by two linear layers. An embedding layer can be seen as a lookup dictionary where each entry represents a specific element, i.e., the weights relative to that same entry. When learning, these same weights are refined, meaning these representations are refined/learned. In this case, for the item embedding layer, each entry in the dictionary contains an item's D-dimensional representation. The following two linear layers have the shape $|I| \times D$. Their purpose is to refine the items' representations further from the previous item embedding layer.

On the other hand, the session encoder $f_{session}$ is also composed of two main sections, an item embedding layer, followed by a sequence encoder that will be described further in section 3.3.3. This was the initially proposed model in [7] that used a metric-learning to tackle session-based recommendation, which we will refer to as DML-Cross-Full, illustrated in figure 3.2.

However, in the same work, they also proposed a slight variant of this cross-modal encoding technique, with changes to the item embedding layers used in both encoders, f_{item} and $f_{session}$. This new variant, instead of having one item embedding layer in each encoder like DML-Cross-Full (fig. 3.2), resorts to having only one single item embedding

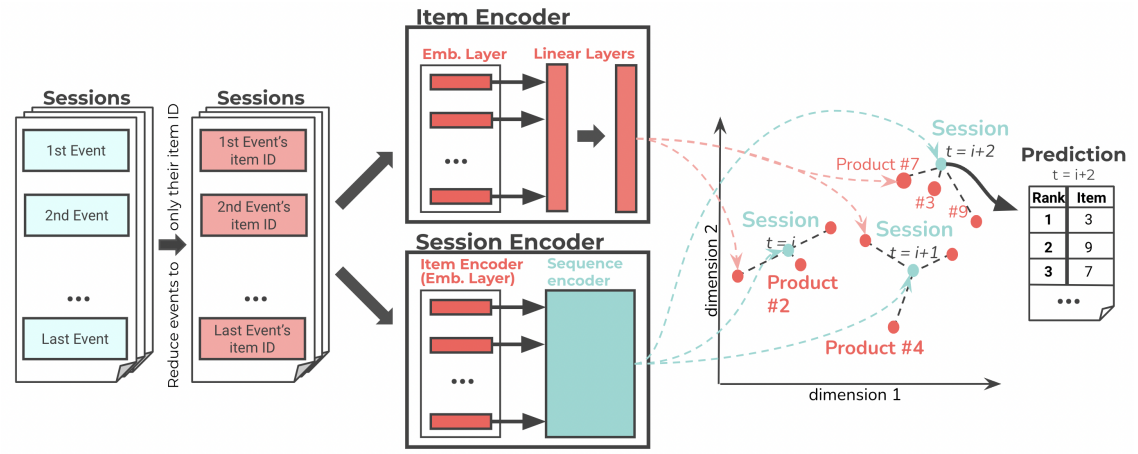


Figure 3.2: Model overview using two separate encoders for items and sessions.

layer that is now common to both encoders. Their ablation study showed that this simpler variant obtained better performance for next-item recommendation than the previous DML-Cross-Full. From now on, we will refer to this simpler model as DML-Cross-Shared.

This difference in performance was not discussed in [7], however, we believe that having separate item embedding layers, one for the item encoder f_{item} and the other for the session encoder $f_{session}$, could hypothetically be helpful if we were interested in learning separate item representations, the items' inter-session representations (item embedding layer of the item encoder), and the items' intra-session representations (item embedding layer of the session encoder). The results obtained in [7] seem to confirm that the more complex DML-Cross-Full with more complex and redundant item representations is only adding extra overhead and noise, worsening the model performance.

3.3.2 Joint Session-Item Encoding

Twardowski, Zawistowski, and Zaborowski [7] reported enhanced model performances after simplifying how items representations are processed in both involved encoders f_{item} and $f_{session}$, compared to the encoding scheme from the previous section 3.3.1. Based on this, we go further and propose an even more straightforward encoding technique focused on the simplification of the item encoder f_{item}.

In the previous section 3.3.1 we have shown that the item encoder f_{item} was comprised of two main parts: an item embedding layer followed by two linear layers with shapes $|I| \times D$. We also explained that the initial item embedding is essentially a lookup dictionary, where, for each item it_j in the dataset $\mathbf{I}$ there is an entry in the embedding layer that contains its representation. This embedding layer is also trained along with the rest of the network, where for the embedding layer, learning essentially redefines the representations it contains per each item. These representations are then fed onto the following two linear layers in a chance to refine such representations further.

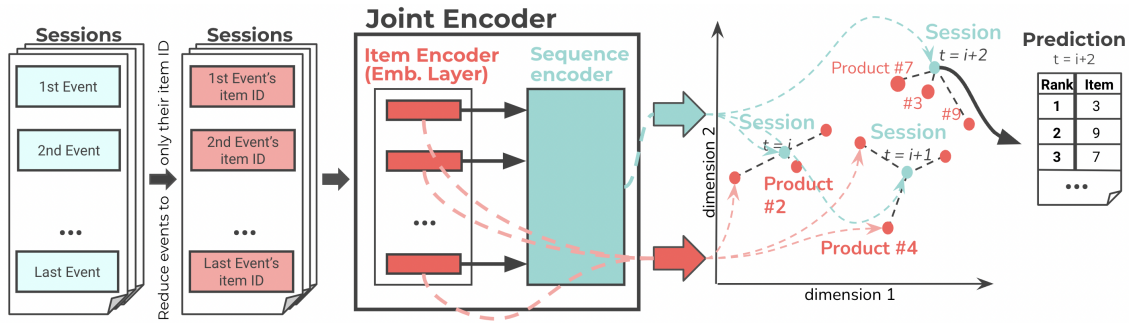


Figure 3.3: Model overview with a single joint-encoder for session and items.

Our contribution here consists in taking the DML-Cross-Shared model and removing the two extra linear layers for the item encoder. This change makes the item encoder consist of only the item embedding layer and has $2 \times (|I| \times D)$ less training parameters to train. We call this model the DML-Joint, and we can see it illustrated in figure 3.3.

3.3.3 Sequence Encoder: TagSpace

In this section we will explain in detail the sequence encoder, the main component of the session encoder $f_{session}$ that comes after the item embedding layer. We can see this illustrated on either figures 3.2 or 3.3.

Twardowski, Zawistowski, and Zaborowski [7] tested their DML-Cross-Shared models using different sequence encoders, like CNN-based or RNN-based encoders. For our work after running some preliminary experiments with the same sequence encoders used in [7], we decided to stick with TagSpace[44] as our sequence encoder.

TagSpace is an efficient CNN-based model, proposed by Facebook. This model is originally intended for the task of hashtag prediction in corpora.

TagSpace was originally intended to predict which hashtags are most adequate given a certain corpus as input. We can easily find similarities between both contexts, where instead here, the input corpus (sequence of words) can be seen as the input session (sequence of events) and the hashtags to predict, the items we want to recommend.

On figure 3.4 we can observe the TagSpace[44] architecture adapted for our use case, with N as the total number of different items in the dataset, H as the number of hidden layers and sl as the fixed length of our sessions.

The choice of this specific architecture for our sequence encoder is due to two main reasons. The first reason lays on the fact that it has shown to attain promising results in challenging conditions where scaling is an important factor [44]. The other reason is that several other works have shown than CNN-based architectures besides being much more scalable than other more complex state-of-the-art sequential models like RNN-based architectures, CNN-based models have also shown to behave well in detecting spatial

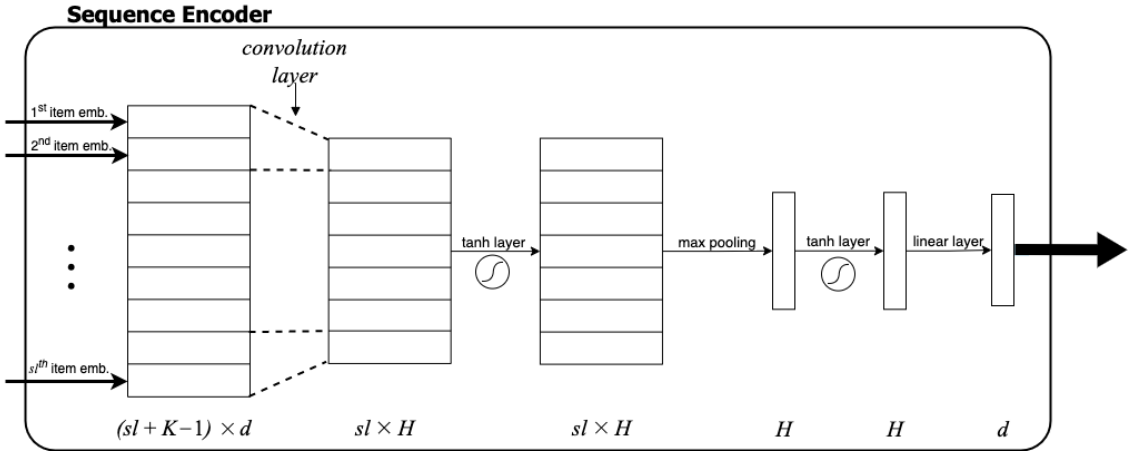


Figure 3.4: Architecture of Facebook's TagSpace, adapted for our domain. D - the dimension of the embeddings to use, as previously defined in table 3.1. sl - the fixed max session length value used, which in the experiments we conduced on chapter 4 we set this to 15. H - the number of hidden layers, in our case we used 256. K - the filter dimension, in our case we used a dimension of 3.

patterns[45], which in our case would mean detecting patterns inside a session, further reinforcing our confidence in using this as our sequence encoder.

Nevertheless, TagSpace[44] has the disadvantage of requiring a fixed session length sl (fig. 3.4), a consequence of its CNN-based architecture. To overcome this, we defined a fixed session length of 15 on the experiments we realized on chapter 4, discarding sessions with more than 15 events and padding with 0's the ones smaller than 15 of length, as in [7].

We believe this to constitute the main disadvantage of the model which could be solved by either using another architecture as capable and fast as TagSpace[44] that does not have this constraint of a fixed length per session, or attenuated with a simpler solution where sessions with higher than 15 of length could be split in splits of 15 length each, as separate sessions, instead of disregarding all of it.

3.4 Session Content-Based Encoding

Content semantic properties are central in determining the true interest of a user in a given item. In the context of a session, not only one should leverage collaborative knowledge from similar sessions to make a recommendation, but it is also important to keep track of the semantic characteristics of the items with which the user interacted in previous time steps. Therefore, apart from modelling user sessions and items relationships, semantic properties of items should account to make better recommendations.

Throughout a session s_i, an anonymous user interacts with multiple items. Our goal

Data Used		Collaborative-Filtering	Content-Based
Session	ID	✓	✓
	Timestamp	✓	✓
Event	ID	✓	✓
	Type		✓
	Time step in session	✓	✓
Item	ID	✓	✓
	Metadata Properties		✓

Table 3.2: Data used in collaborative-filtering vs. content-based recommendation.

is to leverage the two sources of clues present in this interaction for performing a recommendation: a) cross sessions similarity (collaborative aspect) and b) items it_j and event context metadata (content-based aspect). While from a) we obtain sequence item interaction trends, from b) we extract semantically discriminative elements that provide context from each item interaction, which will be crucial for the following recommendation.

To accomplish this, we seek an approach that does not require any modification to existing collaborative learning models but instead biases their learning. Hence, our system is designed to provide the model with prior item semantic and session context so that the collaborative learning phase is not severely disrupted with noise. We divide our contribution into two components: 1) item it_j encoding, which will embed both its properties and capture session context patterns, and 2) an enriched joint session-item common space.

For item representation, we consider two fundamentally different item encoding schemes: discrete encoding (`Item2Discrete`), in which item properties p_k are mapped to a sparse and discrete representation, and embedding-based encoding (`Item2Vec`), in which not only item properties p_k but also the context of the event through the event type identifier along sessions, are mapped to an embedding space learned in an unsupervised manner. Both encoding schemes can easily be used across different recommendation domains.

Once we encode each item it_j in a single fixed representation using either technique, `Item2Discrete` or `Item2Vec`, we initialize these to initialize the weights in the item embedding layer of the DML-Joint model with the item encoders. After this initialization of weights, representing the initial item representations in the embedding layer, training with DML-Joint can start.

3.4.1 Item2Discrete: Discrete Item Property Encoding

In this section we start by introducing our first proposed technique to make use of content-based learning, the `Item2Discrete`. `Item2Discrete` takes as input an item it_j and generates a discrete and sparse representation of that same item, encoding its properties $p_k \in it_j$ in a rather efficient manner.

Given the list of properties p_k of an item it_j, we apply a bag-of-words-like encoding scheme grounded on the assumption that for all items, every property $p_k \in \mathbf{P}$ is defined. Namely, for every distinct property p_k with V different values as the vocabulary size, we create a V-dimensional one-hot-encoding representation. Then, we concatenate all these properties' representations for that item in a single vector effectively representing this item. Given an item it_j, its encoded representation will be the concatenation of the bag-of-words encodings of each $p_k \in \mathbf{P}$, where the most recent attributed value for each property is used. Since the representations are a concatenation of one-hot encoded vectors, the result is a sparse vector representation with 0's and 1's, thus the name of this method, Item2Discrete.

This encoding technique is targeted for categorical data, numerical data or techniques to convert numerical data into categorical is out of the scope of this work. To obtain these bag-of-word encodings we need to get the value each property has for each item to perform the one-hot operations. This assumes that items are not dynamic, in the sense that their properties once defined do not change over time. It is an unrealistic approach, some properties might not ever change (e.g., category), but others can change (e.g., imdb discrete 1 to 10 star rating for movies/shows). As we will see on the experiments chapter 4 this phenomenon happens for the case of the retail rocket dataset. To solve this we derive the unique values of each property per each item from their last state in the dataset, since this describes the products at their most recent state and recency has proven to be a powerful tool in recommendation systems [9].

The concatenation of these bag-of-words encodings per property results in a very sparse vector representation which depends on the vocabulary involved in each property $p_k \in \mathbf{P}$ and the number of properties selected $|P|$.

The computational complexity of the encoding could be summed up as $O((\sum_{p=1}^{P} V(p))+(|I|\times|P|))$, where the first parcel describes the vocabulary building and the second part the encoding of the items itself, and we can see that increasing the number of properties has a linear computational complexity impact on the overall encoding process because the second parcel in charge of the encoding of the items will dominate the overall computational complexity as |I| will tend to be always much higher than any possible vocabulary size of a property $V(p)$. On the other side, if we want to use properties with higher vocabulary sizes this has a higher impact

Such representations are supposed to be added to the embedding layer that encodes the items (figs. 3.5 and 3.6) with D-dimensional weights, but since we are dependent on the choice of the number of properties $p_k \in \mathbf{P}$ selected and their respective vocabulary

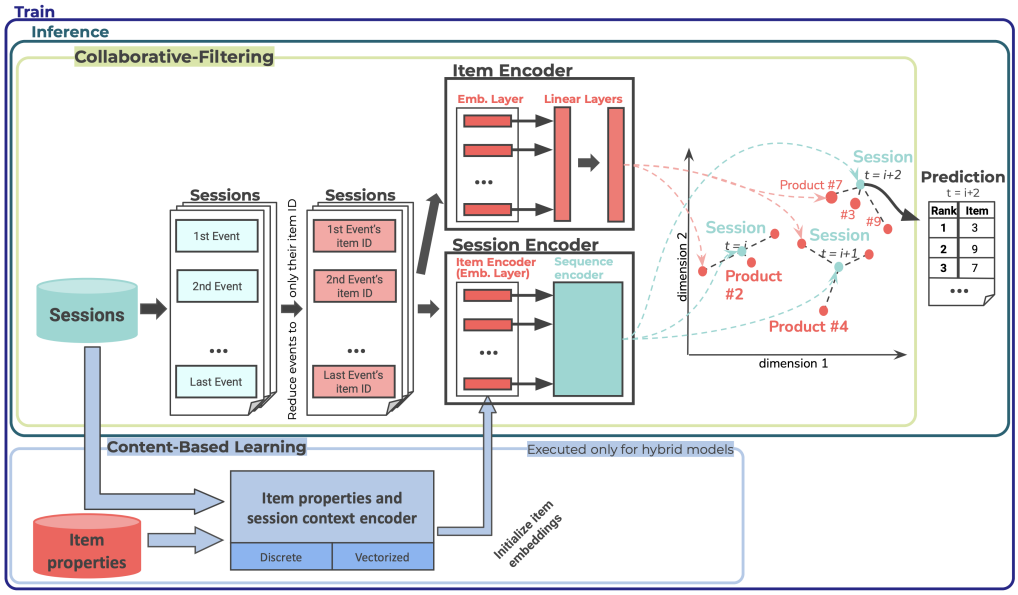


Figure 3.5: Hybrid recommendation cross-modal subspace for items and sessions using separate encoders for items and sessions.

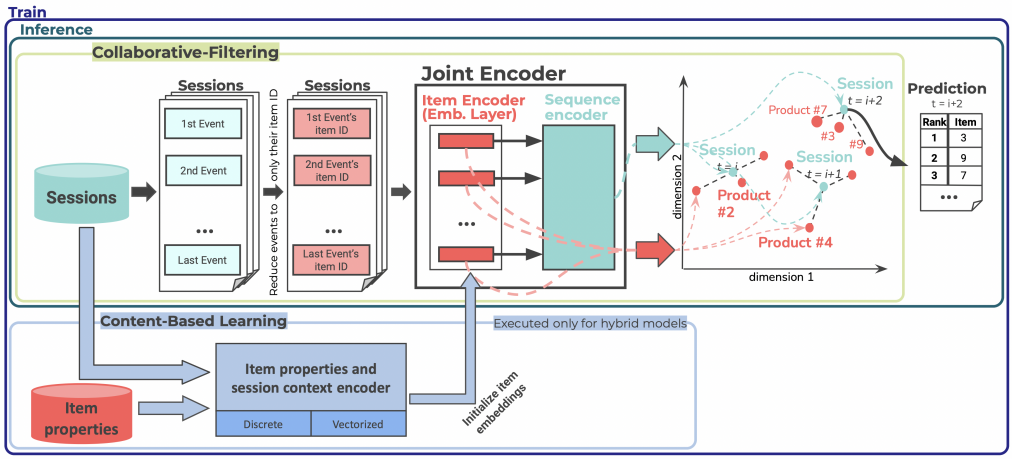


Figure 3.6: Hybrid recommendation with a joint encoder.

technically the choice of the embeddings dimension D is dependent on this circunstances. In the experiments we performed in chapter 4, we restricted the dimension D of these representations to 400, where only the first 400 positions are kept.

3.4.2 Item2Vec: Item Property & Session Context Encoding

In this second item it_j encoding approach, `Item2Vec`, we take a step further by generating continuous representations, thus mitigating the shortcomings of discrete representations.

`Item2Vec` is based on Word2Vec's[23] Continuous Bag-of-Words Model where given in our case a context window of size c, given a certain event token at position t, $event\_token_t$ (eq. 3.6), we take $\frac{c}{2}$-th event tokens before it and $\frac{c}{2}$-th event tokens after, totaling c side

s_{id}	$s_{id,ts}$	item_id	cat_id	ev_typ	string
1	1	6	2	view	item_id@6 categoryid@2 event_type@view
1	2	3	10	view	item_id@6 categoryid@2 event_type@view **item_id@3 categoryid@1 event_type@view**

Table 3.3: Sessions dataset example for the RSC15 dataset. Sessions transformed to Word2Vec friendly format of "words"and "sentences", which is then used as input to train a Word2Vec method and retrieve the learned embeddings of each item

event tokens, whose representations are averaged with the intent of predicting the current event token at position t. This technique has a training computational complexity of $O(c \times D + D \times \log_2(V))$, where V is the vocabulary size dependent on the dataset and its treatment. This technique learns an embedding space in which the structure groups items of similar contexts close together. Namely, given an item it_j, appearing in event e_t^i in a session s_i, its item session context will be captured through a context-window, that will be trained with sequences of session tokens $z_{i,j}$.

Apart from item-specific properties, we also aim at capturing the items' session context, i.e. which items usually precedes or succeeds the item it_j in a session s_i at step t and event context information, i.e., the type of events e_t.

`Item2Discrete`, previously described in section 3.4.1, was not able to deal with changing property values of the items. Instead, now with `Item2Vec` we are able to deal with dynamic content along time, this method is conceptually different from the previous, it accepts the data exactly as it is at that moment in time. This leads to a less noisy representation of items and less manual data handling.

$$event\_token \in \{ID, event\_type \| , \bigcup_{k=1}^{|\mathbf{P}|} p_k^{key} \| @ \| p_k^{value}\} \tag{3.6}$$

Each $z_{i,j}$ is defined as a sequence composed of tokens, obtained from expanding the target item it_j event e_t^i, as well as the surrounding events, according to a context window of size c. Formally, given the set of all items $\mathbf{I}$, the set of all sessions $\mathbf{S}$ and a context-window of size c, the objective is to maximize the average log probability.

$$z = [ID, event\_type, p_1^{key} \| @ \| p_1^{value}, \ldots, p_{|\mathbf{P}|}^{key} \| @ \| p_{|\mathbf{P}|}^{value}] \tag{3.7}$$

Conceptually, given a target item it_j occurring in an event e_i^q, from a session s_i, the model will be trained to maximize the probability of items it_* that belong to events $e_i^{t+\delta}$ and appear within the context-window c, given the item $it_j \in e_i^q$.

To obtain the session sequence Z_i of a session s_i we first need to extract event token sequences for each event $e_t^q \in s_i$. All event token sequences respect the same pre-defined order to maintain consistency. We always have the event tokens for the item identifier

and the type of the event, while tokens relative to the properties vary according to the dataset and the selection of properties made in equation 3.73.6.

$$Z_i = [z_i^1, \ldots, z_i^k] \tag{3.8}$$

Usually for the properties selected, the respective tokens will only consist of the property value p_k^{value} for the k-th property of the event e_i^q of a session s_i, unless word clashing is possible as previously explained. This always results in one token per property. We then concatenate the resulting tokens to an ID and event type (e.g. *click*, *view*) token, resulting in a sequence of tokens for a single event as in equation 3.7. A session can be described by the the concatenation of these sequences obtained from its events, as in equation 3.8 (see table 3.4.2 for an example):

This encoding approach is highly flexible, allowing one to include extra information in each z_i^q as desired. The amount of session context to be included in the item embedding can be controlled through the magnitude of the context-window c. The fact that the item encoding can be learned in a completely unsupervised manner enables it to be used in a myriad of recommendation settings. In terms of embedding representation capacity, for highly complex scenarios with a large number of properties per item, one can tune the embedding size appropriately. While categorical data can be easily integrated, numerical values still need to be discretized. We leave more thorough exploitation of numerical properties and their role in an item context for future work.

3.5 Optimization

3.5.1 Loss Functions

In metric-learning, our goal is to make distances between elements, i.e., their projections (embeddings) in a certain space, equivalent to the dissimilarities between them. To accomplish this, we use a loss function that is capable of guiding a neural network model in building a space with such characteristics. We use the widely known Triplet-Loss and a more recent approach used in several works [7], the SDML Loss. These provide very different approaches for metric learning, an interesting subject of research which we will address further on this work in chapter 4.

3.5.1.1 Smoothed Deep Metric Learning (SDML) Loss

SDML was proposed by [40] for a Question Paraphrase Retrieval system implemented as a Neural Information Retrieval system, with the goal of providing a better alternative for noisy scenarios when compared with the incumbent triplet-loss [40], and also later used for session-based recommendation in [7]. This loss function is different and aimed to replace the incumbent triplet-loss for more noisy scenarios.

It transforms the distance d of the projection of a session $s_k \in S$ using the session encoder φ and the projection of the positive and negative sampled items in set Z obtained

from the encoder ω into probabilities using a softmax applied over the negative squared euclidian distance (eq. 3.9),

$$p(it_j|s_k) = \frac{exp(-d(\varphi(s_k), \omega(it_j)))}{\sum_{it_j \in Z} exp(-d(\varphi(s_k), \omega(it_j)))}, \tag{3.9}$$

Combining the previous probability $p(it_j|s_k)$ (eq. 3.9) with $p'(i)$ (eq. 3.10),

$$p'(it) = (1-\epsilon)p(it) + \frac{\epsilon}{N}, \tag{3.10}$$

we can compute the final loss using the Kullback-Divergence between them (eq. 3.11)

$$L_{SDML} = \frac{1}{|S|} \sum_{s_k \in S} KLD(p(it|s_k) \,\|\, p'(it)), \tag{3.11}$$

a smoothed probability distribution of the true labels of the sessions in S and the probability distribution obtained from equation 3.9.

To obtain a lower loss, the Kullback-Divergence Loss between the smoothed real probability distribution of the sessions to each item and the probability distribution obtained from the distance of sessions to these same items on our learnt space should be as low as possible, evidencing a lower entropy between our model's predictions and the ground-truth. These distances are calculated pairwise between the session s_k and its sampled positive and negative items in Z.

This loss function uses a smoothing technique that makes it less rigid than the triplet-loss, therefore it is more suitable as a regularization tool to prevent model overfitting and robust in noisy scenarios [40].

3.5.1.2 Triplet-loss for Session-based Recommendation.

During a session, at iteration t, the goal is to learn a recommendation embedding space in which items will be close to each other if they are semantically similar and have high relevance towards being recommended.

$$l_{Triplet}(x_a, x_p, x_n) = max(0, d(x_a, x_p) - d(x_a, x_n) + m) \tag{3.12}$$

Formally, we form triplets (x_a, x_p, x_n), where a session embedding is the anchor element x_a, and a product is the positive element x_p if it was selected, or the negative element x_n otherwise. The triplet-loss attempts to minimize the distance between the session embedding (the anchor) and selected product embedding (the positive element), while simultaneously maximizing the distance between the anchor and negative sample (the ignored or discarded product). A fixed margin is used to constrain the difference between the anchor-positive and anchor-negative distances, where, the anchor-negative distance should be at least m distanced from the anchor-positive.

In our case the anchor element is always a session embedding where the positive and negative elements are always items' embeddings, resulting in the following equation 3.13:

$$l_{Triplet}(s_a, it_p, it_n) = max(0, d(s_a, it_p) - d(s_a, it_n) + m) \tag{3.13}$$

where it_p is a positive item, and it_n a negative item, relative to our current session s_a. Some works propose the use of adaptive margins [41, 42] resulting in a more flexible loss function better representing this subspace, where even such such adaptive margins could be guided by temporal traits [42] which play an important role in our use case through recency. We believe this last point to be an important research direction that we leave for future work.

3.5.2 Sampling: Positive-Negative Sampling

To train in a metric-learning context, we need to sample positive and negative samples (items), and the choice of the sampling technique has shown to play an important role in metric learning[7, 43]. We apply the same sampling technique of [7], given the results they obtained on their ablation study. In this technique, sampling occurs before each epoch, and sessions are randomly sampled and split into two parts: (a) the events/items before the split position that will be the input; and (b) the next *pos* events/items after the split position, which make the ground-truth. The events/items after (ground-truth) are our positive samples. In contrast, the same number of negative samples/items are randomly sampled items from the dataset **I** where the probability distribution is set by the frequency of such items in the train split, i.e., where more popular items have a higher chance of being sampled for this task.

3.6 Temporally Smoothed Recommendation

Other works[7, 9, 10] have raised awareness to the importance on incorporating temporal characteristics in learning. In this section we base on this idea to present two different techniques that take advantage of this. The first one was proposed by [7] and resides on the usage of a variant of triplet-loss that takes advantage of temporal proximity, the second one is proposed by us, and consists in the usage of a temporal re-ranker that incorporates recency for the DML models.

3.6.1 Weighted Triplet-Loss

The original triplet-loss formulation does not take in account different degrees of relevance for the items in the ground-truth, i.e., for a given session s_i the model considers equally relevant the items inside the ground-truth. Therefore it is learning a space where such items would be equally distant to s_i, although one could argue they are not equally relevant and by consequence they should not be equally distant. On one hand this could provide to be beneficial as a form of regularization because in a session-based setting, typically such interactions are very close in time and having different degrees of relevance might not be useful, but on the other hand in certain contexts (e.g., longer in time

sessions) this behavior might have a negative impact. In order to answer both cases a temporal-proximity factor is taken into account in the triplet-loss formulation, an idea proposed by Twardowski, Zawistowski, and Zaborowski [7]. This applies the following weighting function $\sqrt{1/(1+z)}$ being applied on the positive samples distances, with z being the position of the positive item in the ground-truth. This weighted triplet-loss is formulated below on equation 3.14,

$$l_{Triplet}(s_a, it_p, it_n, z) = max(0, d(s_a, it_p) - d(s_a, it_n) + m) \times \sqrt{1/(1+z)} \tag{3.14}$$

Since this gives more weight for the loss computation regarding sampled triplets involving the initial positive samples in the ground-truth, triplet-loss is not being smoothed with temporal proximity.

3.6.2 Temporal Re-Ranking

Session-based nearest-neighbors (SKNN) ranks according to scores obtained through item co-ocurrence in sessions and this simple method has proven to be a superior to much complex approaches like GRU4REC[27]. Later improvements based on top of SKNN were proposed, like VSTAN[10] and VSKNN[9] further strengthening how strong such methods behaved when benchmarked against more complex methods.

The combination between SKNN and the complex RNN-based GRU4REC has shown that it is possible to obtain a superior model by combining a simple session k nearest neighbors algorithm and a complex neural-network aimed at the same goal of session-based recommendation albeit being fundamentally different. Our idea is to do a similar technique, but now, combining our model with the more powerful VSKNN instead of SKNN, as it incorporates temporal traits and has been proven to be superior to SKNN shown by Twardowski, Zawistowski, and Zaborowski [7] and Ludewig and Jannach [9]

These same works have also shown that the temporal aspect plays an important role in session-based recommendation.

VSKNN[9] is a well-established collaborative-filtering session k nearest neighbors algorithm for session-based recommendation that takes advantage of temporal recency as a smoothing factor compared to its predecessor SKNN[8]. First we define below how VSKNN works:

Given a session s_t, let the set of neighbouring sessions be $N_{s_t} = \{s_j : it_k \in (s_t \cap s_j), j \neq t\}$. The session similarity function is defined as:

$$sim(s_t, s_j) = \frac{1}{|s_t|} \cdot \sum_{it_k \in s_t, it_k \in s_j} w_{it_k} \tag{3.15}$$

where $|s_t|$ is the number of items in session s_t, and w_{it_k} weights the item it_k based on its position using a smoothed decaying function that assigns more weight to items it_k that appear later in the session.

Finally we will compute the score for every item $it_k \in N_{s_t}$ and recommend the items ordered by their respective score:

$$score(it_k; s_t) = \sum_{s_j \in N_{s_t}, it_k \in s_j} [sim(s_t, s_j) \cdot (1 + idf(it_k))] * t_{sim}(s_j, s_t)^2 \tag{3.16}$$

where $t_{sim}(s_j, s_t)$ does the temporal smoothing, applying a weighting function to the most recent position where a common item occurs in the other session s_j, giving the highest weight to a match on the last item of s_j, i.e., the most recent. In our implementation we used a logarithmic weighting function for t_{sim}.

We propose a novel model where the results given by our DML-Joint model are re-ranked by VSKNN. This results in a meta model that takes advantage of the powerful session k nearest neighbors technique and the expressiveness of a metric learned subspace representing both sessions and items where temporal traits are also taken in account through VSKNN.

First, for a given session s_t at step t, retrieve the items rank $I_t = it_0, \ldots, it_k$, where k is the rank size, from the neural network model with metric learning. This is obtained by simply computing the distance between the current session s_t to every item in the dataset $it_k \in \mathbf{I}$. We define each of these distances between the session s_t and an item it_k as $DML_{dists}(s_t, it_k)$ that defines the similarity between the current session s_t and each item it_k, based on the learned cross-modal space. For performance reasons, after computing $DML_{dists}(s_t, it_k)$ we compact this ranked list to only the top 200 items. These distances are then converted to scores for compatibility purposes when re-ranking with VSKNN, with the closest items with lower distance having the highest scores. VSKNN does not encompass the concept of distances when measuring similarity, so we need to convert distances to similarity scores from DML using the following equation 3.17,

$$DML_{score}(s_t, it_k) = \frac{1}{1 + DML_{dists}(s_t, it_k)} \tag{3.17}$$

The DML_{score} and $VSKNN_{score}$ results are both normalized between a range of 0 to 1 during this procedure as their values do not set in the same ranges.

$$\begin{aligned} score(s_t, it_k) = (1 - \lambda) \cdot normalized(DML_{score}(s_t, it_k)) \\ + \lambda \cdot normalized(VSKNN_{score}(it_k, s_t)) \end{aligned} \tag{3.18}$$

Finally re-ranking occurs following equation 3.18, where a higher λ will result in a higher re-ranking power by VSKNN.

3.7 Computational Complexity Analysis

Our focus must go beyond which methods and techniques are the best overall but also inspect in what use cases each is more adequate.

Neural Network models' computational burden in inference is usually not a concern if we compare it to their training phase computational cost. However, we specially consider computational complexity for inference to be a decisive factor in the context of session-based recommendation.

Below we can find the computational complexity of VSKNN for the inference phase in equation 3.19:

$$
\begin{aligned}
&O(2\times|I|+3\times|S|+|session\_items|+sample\_size+min(|session\_items|,|other\_items|)+\\
&sample\_size\times log(sample\_size)+k\times(|session\_items|+2\times|other\_items|+\\
&(k\times|other\_items|)\times log(k\times|other\_items|)\Leftrightarrow\\
\Leftrightarrow &O(2\times|I|+3\times|S|) \quad \text{since} \quad |S|>>|sample\_size|,k \quad \text{and} \quad |I|>>|session\_items|,|other\_items|\\
\Leftrightarrow &O(|I|+|S|)
\end{aligned}
\tag{3.19}
$$

, where k and sample_size are both VSKNN hyperparameters used to limit the search of possible candidate session neighbors.

Now below we can also see the computational complexity for the inference phase of our proposed DML-Joint model in equation 3.20:

$$
\begin{aligned}
&O(3\times|session\_items|+2\times|I|)\Leftrightarrow\\
\Leftrightarrow &O(|I|) \quad \text{since} \quad |I|>>|session\_items|
\end{aligned}
\tag{3.20}
$$

For VSKNN, this inference operation is quite more complex as most of the algorithm computations occur on this phase. The computational complexity is described in equations 3.19 and essentially its computational cost is proportionally dependent on the number of items $|I|$ in the dataset plus the total number of sessions $|S|$. We should also note that $|S| >> |I|$ in this field, which in some cases like in one of the datasets that we used in chapter 4 the number of sessions can even reach up to 10 times more than the number of items in the dataset. It is now clear why VSKNN uses the hyperparameters k and sample_size to control the search of possible neighbor sessions in the dataset.

In the case of the DML-Joint models without temporal re-ranking, for inference the cost is essentially dependent on $|I|$, as we can see on equation 3.20. The main reason for this is because the ranked item list results from a distance computation between the projection of the current session against all items in the dataset.

After gathering the computational complexities of both models, we can also say that the DML model that uses VSKNN as a re-ranker on top has consequently the same computational complexity as VSKNN for inference.

4

Experiments

In this chapter, we present the experiments conducted regarding each of the techniques and contributions described in the previous chapter 3. We first present the datasets selected, expose their main characteristics, explain the data preparation steps conducted and finally, analyze the datasets regarding their session length and item frequencies. We also study in detail the experimental methodology, more precisely the metrics to evaluate the models, given the common occurrence of flaws in other works in this field [39]. Our experiments focus on determining the usefulness of our contributions by comparing our models to other state-of-the-art models in the field. We analyze the impact of the session length in the performance on our models and other baselines and discuss each model's use-cases based on asymptotic analysis focused on the inference phase.

4.1 Datasets

The datasets used in this chapter are two openly available datasets, the Retail Rocket (RR) [46] and the ACM 2015 Recsys Challenge (RSC15) [47] datasets. The RR and RSC15 datasets contain users' interactions over items in an online retailing context.

ACM 2015 Recsys Challenge (RSC15) dataset The RSC15 [48] dataset was made available for the ACM RecSys Challenge 2015[1] by YOOCHOOSE[2], a recommender system as a service company. The dataset was collected from an (anonymous) European online retailer in 2014, containing data over six months. The contents are hashed due to privacy concerns. The dataset is a set of sessions in which each session contains mainly the click events that the user performed in that session. Sometimes, a session can contain a buy event, as a final event, if the user buys something, therefore terminating the session. This dataset was released to incentivize the research of recommender systems that could predict what item the user wants to purchase, while our goal here is to provide relevant recommendations for the users.

[1] https://recsys.acm.org/recsys15/challenge/
[2] www.yoochoose.com

Table 4.1: RSC15 Dataset Description

Property	Description
Session ID	Unique identifier of the session
Timestamp	When the event occurred in the format YYYY-MM-DDThh:mm:ss.SSSZ
Item ID	Unique identifier of the item
Category	Character representing the context of the event. This was not used.

Retail Rocket (RR) dataset The RetailRocket (RR) [46] dataset was collected from real-world e-commerce websites, raw data, with values hashed for privacy reasons. User interactions are of type *view*, *add-to-cart* or *transaction*.

Property	Description
Category ID	Unique identifier of a category of items
Parent Category ID	Unique identifier of the parent category if there is any
Item ID	Unique identifier of the item
Item Property	Item Property identifier can be a number (from the hashed original property) or just either categoryid or available
Item Value	The value of this item property
Timestamp	Timestamp of the property, in the UNIX format, as it can change over time
User ID	Unique identifier of the user
Timestamp	When the event occurred in the UNIX format
Item ID	Unique identifier of the item
Event Type	"view", "addtocart"or "transaction"
Transaction ID	Identifier of the transaction if this was one.

Table 4.2: RetailRocket Dataset Description

Neither dataset contains events that explicitly request the exclusion of an item (e.g., dislike a particular item) or a set of items in the search space. In this absence of negative feedback, we view all events in both datasets as positive events, i.e., as valuable cues for the subsequent recommendation.

4.1.1 Data preparation

For each dataset, and following relevant works [7, 10, 27, 49–53], we only consider a subset of the same: first by considering only the n-th last split (more recent) of the datasets, and secondly through filtering, because a significant quantity of data is involved, and training on only the most recent fractions has shown to yield better results[53]. We apply the following filtering conditions for both datasets: (a) items must occur at least five times in the dataset, and (b) sessions must have more than 1 event but at most 15. The reason for (b) stands due to the fact that at least a session needs to contain 2 events so that learning

Table 4.3: RR and RSC15 Dataset Statistics - (original) and their nth last split with sessions that have more than one event and items with at least 5 occurrences.

Dataset	Items	Sessions	Events
(RR) RR-5	(208k) 40k	(1.059M) 225k	(2.025M) 761k
(RSC15) RSC15-64	(53k) 8k	(9M) 145k	(33M) 429k

can happen, the first event being the input and the seconds event being the ground-truth, but sessions cannot exceed 15 events in length because in the sequence encoder we use the TagSpace CNN-based architecture (explained in section 3.3.3) which fixes sessions at a length of 15, padding the ones smaller with 0's and discarding the ones that exceed that length. We apply iterative filtering until both these constraints are valid.

In the case of RSC15, we only consider the 64-th contiguous in time split of ACM RecSys Challenge 2015 dataset to use as in [7], and sessions are already anonymized. More specifically, no user information is available in this dataset, only sessions.

For RR, following [7], we only use the 5-th contiguous in time split of RetailRocket. However, the dataset contains all interactions of users without any session aggregation. Instead, they come initially aggregated by user, so we transform this session-aware dataset into a session-based dataset by transforming each identified user session into an anonymous session. To achieve this, we assume a session timeout of 30 minutes to partition sessions of the same user in different anonymized sessions (session-aware → session-based).

We will refer to the RSC15 and RR datasets as RSC15-64 and RR-5 respectively to differentiate these subsets from the original dataset. Table 4.3 shows the statistics of each dataset concerning items, users, and interactions, before and after splitting and filtering.

For our Item2Disc method previously proposed in section 3.4.1, we require a unique item representation a priori for all the items per dataset, so we analyzed and selected the best candidate features to be one hot encoded as the initial embedding representations of the network. For the RSC15-64 dataset, we only had the *categoryid* property that never changes over time. For the RR-5 dataset, there are many properties with varying levels of noise. To mitigate this phenomenon, we applied a heuristic to filter noisy properties. These needed not to have the same value for all properties, to have values for all items as filling missing values is out of the scope of this scape, not to change more than an average of 5 times per item, and do not consist of the 'available' property. Of the remaining properties, we selected only the categorical properties, obtaining in the end only the *364* and *categoryid* properties.

4.1.2 Dataset Analysis

This section analyzes the datasets after the preparation steps described in the previous section. We focus our analysis on the frequency of sessions per session length (number of events) and another analysis to observe the item popularity.

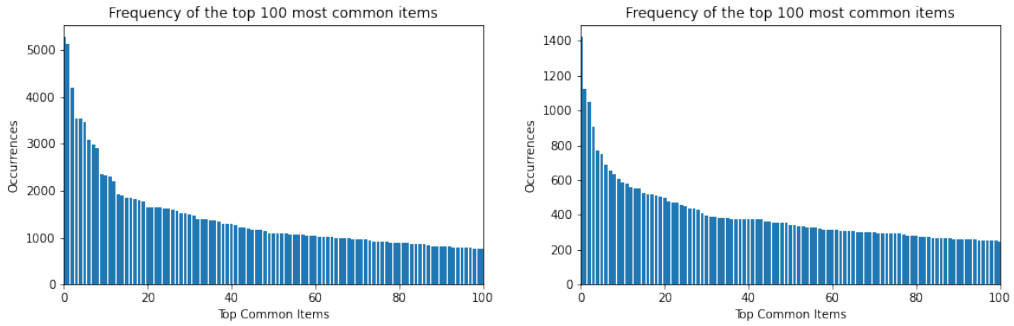


Figure 4.1: Frequency Distribution of items for RSC15-64 (left) and RR-5 (right).

In RSC15-64 and RR-5, the items follow a long-tail distribution as seen in figure 4.1, a characteristic of datasets of the recommendation field and a commonly recognized challenge [54, 55]. This characteristic is also present in the session lengths on both RSC15-64 (fig. 4.2) and RR-5 (fig. 4.3) datasets, that besides following a long tail distribution, it interestingly follows in part a Zipfian distribution where the number of sessions with a certain session length l is double the number of sessions on the dataset with lengths $l+1$, i.e., that have one more event in length. In closer inspection, using a logarithmic y-axis, at the bottom of figures 4.2 and 4.3, we can verify that it does respect Zipf's law for lower session lengths, but that is not the case when we start having longer session lengths.

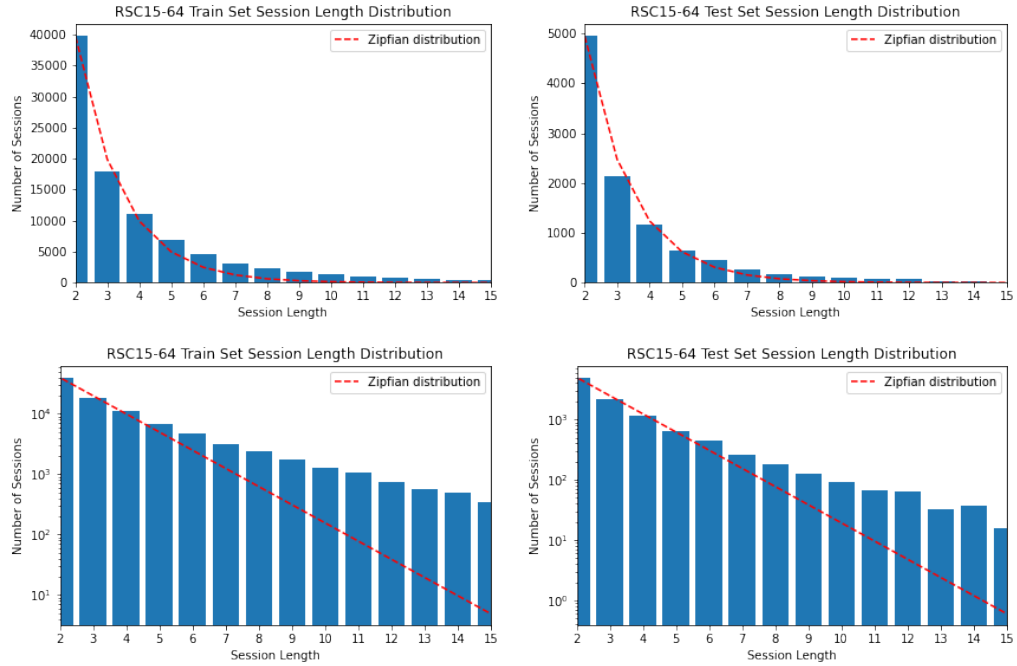


Figure 4.2: Analysis of the session length distribution for the RSC15-64 dataset splits, the train split (left) and the test split (right)

Given our analysis, we expect the models to perform significantly better for shorter

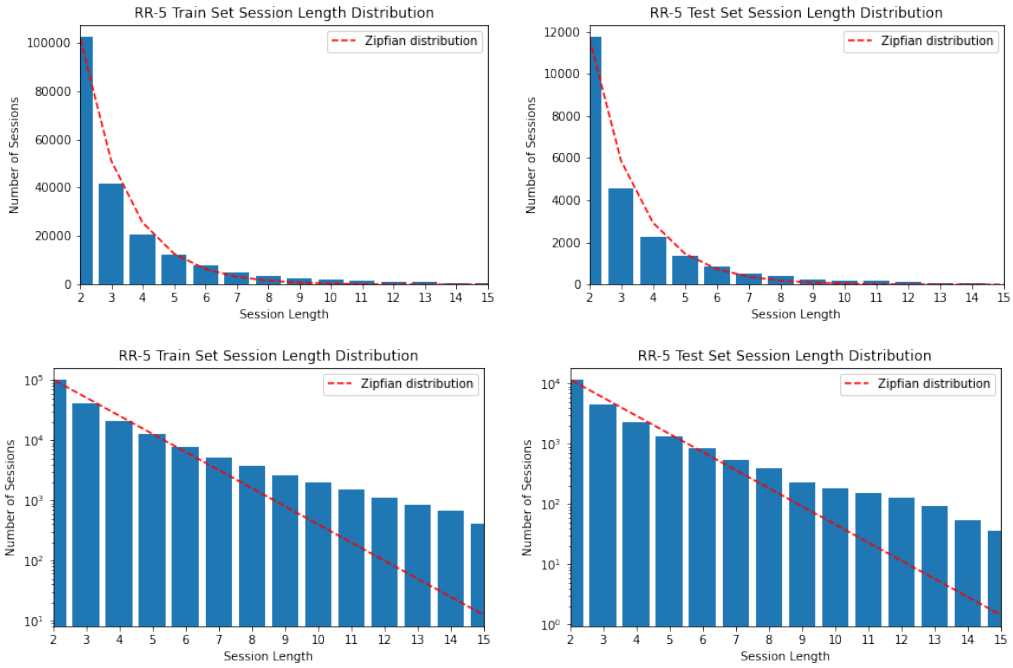


Figure 4.3: Analysis of the session length distribution for the RR-5 dataset splits, the train split (left) and the test split (right)

sessions due to the imbalance observed in the distributions of figures 4.2 and 4.3 and models that learn or can take advantage of the item popularity concept will also tend to obtain better results overall given the popularity-bias evident in figure 4.1.

4.2 Protocol

We start by first splitting each dataset, by using 90% of session interactions as train data and the remaining (i.e. subsequent interactions) as test data. Of this 90% training split training, 5% are reserved for validation. These splits all respect the same distribution in all datasets, which we can observe on figures 4.3 (RR-5) and 4.2 (RSC15-64) for the case of the RSC15-64 dataset.

We resort to data augmentation to further strengthen evaluation conducted on the test set. For this step, we split each session in the test set into multiple test samples for evaluation. For each session in the test set, s_i with length l, we can use it to generate a total of $l-1$ samples for evaluation. This data augmentation procedure is done by splitting it into all possible splits that comply with the following constraints:

- Maintain the same order of the events.
- Contain at least one event for our input split.
- Contain at least one event for the ground-truth split.

- The sum of both input and ground-truth splits must equal the length of the original session.

To illustrate this process, given the following session in our test set $s_i = \left[e_1^i, e_2^i, e_3^i\right]$, with a length l of 3, we can use it to generate the following $l - 1 = 2$ test samples:

$$\begin{aligned} sample1: \quad & input \Rightarrow \left[e_1^i\right] \quad & groundtruth \Rightarrow \left[e_2^i, e_3^i\right] \\ sample2: \quad & input \Rightarrow \left[e_1^i, e_2^i\right] \quad & groundtruth \Rightarrow \left[e_3^i\right] \end{aligned} \tag{4.1}$$

,

4.2.1 Metrics

Recent works [10] have raised awareness of common methodology flaws in the session-based recommendation field and their contribution to the phenomenon of phantom progress[10, 11], where many claimed results and contributions could not be reproduced or, when reproduced, do not validate such claims. Along with our work, we encountered such inconsistencies ourselves. In some works [7, 52] their implementation of the metrics did not seem to respect the definition of well-established literature in this field [56, 57]. We dedicate this section to disclose our interpretation of such metrics following official standards while promptly explaining other variants that we used. Our goal is to promote the reproducibility of our work and prove that the contributions contained in this work are not just phantom progress[10, 11] but are instead reproducible contributions.

We divide our choice of metrics into two sets: (a) the k-agnostic metrics and (b) the @k (at-k) metrics. The set (a) is only composed by a single metric, R-Precision, while the remaining belong to set (b) and requires us to specify a certain k.

Using metrics in (b) helps evaluate the model's predictions towards a specific k cut-off of the return ranked prediction list. This is particularly useful in scenarios where we restrict the shown predictions to a user, e.g., the interface can only show the top-k predictions. If we know the ideal k for our scenario beforehand, the best choice is to opt for metrics in (b) as our main metric(s).

Usually, the chosen k is 1, 10, or 20, with 20 being the most common k used. We believe 20 to be a good choice for k, and the fact that it is the one most commonly used allows our results to become comparable to a more range of other works in this field.

Musgrave, Belongie, and Lim [39] work is very critical of the methodologies and protocol followed by other works in this field that often claim false progress in this field, phantom progress[10, 11], based on unfair comparisons between models tested on different setups or models that cannot be reproduced with the same results. The authors propose that future work follow specific practices that ensure fair comparisons and reproducibility. They recommend the usage of the k-agnostic metric R-Precision in combination with MAP@k, where R-Precision has the advantage of not requiring to fix a certain k, but its main weakness, not accounting for the ranking of correct retrievals, is balanced with the usage of MAP@k.

Ground Truth We consider all the following events at a session split to be the respective ground-truth. In our interpretation, in a session-based context, events inside a session are usually relatively close in time, so restricting our ground-truth to only the next event as in other works [7] is not the best approach. To start, we define the following variables relative to the ground-truth:

$$
\begin{aligned}
rel_s(i) = \begin{cases} 1, & \text{if i-th element in the rank} \in gt_s \\ 0, & \text{if i-th element in the rank} \notin gt_s \end{cases} \\
gt_s = \text{ground truth of session } s \\
N_{gt_s} = \text{size of the ground truth of session s}
\end{aligned} \tag{4.2}
$$

However, as we will explain further, we also compute the standard MRR@k and HR@k metrics for next-item prediction to evaluate the models' performance additionally for such use-cases. For these variants, gt is considered to be the immediate next item, i.e., gt is reduced to its first item.

Below we will explain in detail the metrics that either are open to this ambiguity issue, following the official USA NIST TREC EVAL standards[3] except in some cases where we use a non-standard variant that we also explain in detail and the rationale behind it.

4.2.1.1 PREC@k - Precision at k

Precision refers to the ratio of relevant items predicted among all predicted items. In the case of Precision@k this comes with a caveat. We are forcing the algorithm to give us k predictions. However, in some cases, the algorithms might return less than the k requested predictions, leading to two possible solutions: (i) consider the number of predicted items to be only those that the algorithm indeed predicted or (ii) consider that the algorithm made all k requested predictions, where the remaining predictions not realized consist of non-relevant recommendations. Both possible interpretations lead to very different results, in (i) the metric tends to favor more stubborn algorithms that refuse to give the k requested predictions, while (ii) penalizes such behavior.

We also follow the official NIST TREC EVAL standard where they consider the remaining not returned predictions from the k requested to count as predictions made by the algorithms that consisted in a miss, following approach in (i), which strongly penalizes the algorithms that do not return the k requested predictions. We formalize this interpretation in equation 4.3. The other interpretation (ii) is only viable in what the standard refers to as the Set Precision (eq. 4.4, where the metric is computed only regarding the returned predictions without any penalization, but it requires the metric to be k-agnostic.

When the algorithm returns zero predictions, the precision value is 0 [56].

[3] https://github.com/usnistgov/trec_eval

Precision@k This metric follows the interpretation in (i), it requires us to set a k, filling in with non-relevant hits the remaining predictions not returned by the algorithm up to k. It penalizes models unable to meet our request of k predictions.

$$\textbf{P@k} = \frac{1}{|S|}\sum_{s}^{S}\frac{1}{k}\sum_{i}^{|P|} rel_s(i) \tag{4.3}$$

Set Precision This metric is a k-agnostic precision variant. Since we do not define a k here, we are only concerned about all the predictions actually returned by the number.

$$\textbf{Set\_Precision} = \frac{1}{|S|}\sum_{s}^{S} Set\_Precision_s \tag{4.4}$$

$$Set\_Precision_s = \begin{cases} \frac{1}{|P|}\sum_{i}^{|P|} rel_s(i), & \text{if |P| > 0} \\ 0, & \text{if |P| is 0} \end{cases} \tag{4.5}$$

In some works that happen to openly release their implementation [7, 52], we noticed they did not follow this standard, but instead follow the interpretation (ii). This does not conform to NIST's alternative Set Precision (eq. 4.4), as their interpretation is not k-agnostic, therefore they use a non-standard computation of Precision@k.

We use the Precision@k of interpretation (i) that strictly follows the NIST's standard in our work.

4.2.1.2 HitRate@k

This metric is fairly simple and encompasses only the two following cases for each sample: it is 1 if on the top-k prediction list there is at least one relevant item or 0 otherwise. The result is the mean of these hit rates over all samples. We also additionally compute this metric in a variant where we consider the ground-truth to be only the next exact interacted item as done with MRR@k.

4.2.1.3 R-Precision

This metric has the advantage over the others that it is agnostic of a k. R-Precision equals the number of relevant items in the ranked list up to an R-th prediction. R can vary per each sample as it is equal to the length of the ground-truth gt_s (eq. 4.6) of a certain session. R-Precision is essentially Recall at the R-th position and is usually highly correlated [56] with MAP.

$$\textbf{R\_Precision} = \frac{1}{S}\sum_{s}^{S}\frac{\sum_{i=1}^{N_{gt_s}} rel_s(i)}{N_{gt_s}} \tag{4.6}$$

This k-agnostic metric is immune to changes in the length of the ground truth gt_s which is the case in our work. Since the k-dependent metrics require us to fix a certain k, they under-report when the ground-truth does not fit inside the k requested predictions

$k < gt_s$ while that is not an issue with R-Precision. This metric is also currently used in the TREC EVAL tool.

4.2.1.4 MRR@k - Mean Reciprocal Rank at k

MRR@k is useful in cases where we are only worried about predicting the next event because we either: (a) have only one correct answer (ground truth of length 1); or (b) are only interested in the top-ranked correct prediction. This metric focuses on the reciprocal rank of the highest-ranked correct prediction. Instead of the original formulation, we use a modified MRR@k metric that computes the sum of the reciprocal ranks of each relevant answer, i.e., the sum of the reciprocal ranks of all those relevant events predicted up to the top-k predictions. We do this because this variant better suits our problem, where now we worry more about the overall ranking list than just the highest correct answer, similar to MAP@k. This is the only metric we use on our work that does not have an upper limit of 1.

$$\textbf{MRR@k} = \frac{1}{S}\sum_{s=1}^{S} RR_s@k \tag{4.7}$$

$$RR_s@k = \sum_{i=1}^{k} \frac{1}{i \times rel_s(i) + \epsilon} \tag{4.8}$$

Although not our primary focus, we additionally compute MRR@k for next-item prediction, where only the next following event comprises our ground-truth.

4.2.1.5 MAP@k - Mean Average Precision at k

Currently the most used metric in the TREC community given its good discrimination and stability. This metric is the mean of the average precision (eq. 4.10) across all samples, shown in the following equation 4.9,

$$\textbf{MAP@k} = \frac{1}{S}\sum_{s}^{S} AP_s@i \tag{4.9}$$

$$AP_s@k = \frac{\sum_{i=1}^{k} P_s@i \times rel_s(i)}{N_{gt_s}} \tag{4.10}$$

This metric is useful when we are concerned about the overall ranking of the predictions, promoting more stable recommendations overall.

Musgrave, Belongie, and Lim [39] also proposed the usage of a MAP variant that is k-agnostic called MAP@R, where R is the ground-truth length gt_s, claiming to be better than R-Precision when the goal is to have a k-agnostic measure that is more adequate to evaluate the overall ranking list. MAP@k is still the dominant and the most widely accepted metric across the TREC EVAL community [56]. MAP@k has been widely tested in the research community, and its behavior is well-known. The same cannot be said

regarding MAP@R, which was even removed from the TREC EVAL tool. Given these reasons, and since we already chose to use R-Precision, we decided to use MAP@k in lieu of MAP@R.

4.3 Implementation Details

All experiments were conducted using PyTorch, with 100 epochs and a batch size of 32. An Adam optimizer is used and set for an initial learning rate of 0.001 but controlled by a learning rate scheduler that reduces the learning rate when the primary metric (R-Precision) stops improving. The embeddings dimension used was 400, and for the sampling method at each training step, two sets of items, eight positives and eight negatives, are sampled using the session positive-negative sampler described in section 3.5.2. Specifically for the methods that use Item2Vec, a context-window of size 8 was used.

Given the high number of possible hyperparameters combinations and the computational burden of the DML models, hyperparameter tuning is out of scope of this dissertation.

4.4 Initial Assessment of Nearest Neighbours and Frequency-based Approaches

Our experiments start with more straightforward but powerful methods that either apply a sort of session k nearest neighbors or use other standard frequency-based techniques. These methods are RND, POP, SPOP, MARKOV-1, SKNN, and VSKNN. Although they are not our primary baselines, they are fundamentally different, focused on taking advantage of different features which provide essential cues for our work: some focus on item popularity (POP and SPOP); others on only item co-occurrence in sessions in a collaborative-filtering approach (SKNN and VSKNN); others take into account the temporal characteristics in the dataset (VSKNN); or view the problem as a transition of states where a previous state is believed to influence the following state (MARKOV-1). Some works [2, 7, 9, 58] have proved them superior to many other recently proposed more complex methods, even when compared to neural network-based models and our goal here was to see which strategies stood out to inspire our work.

RND A random recommender with the intent of corroborating the validity of the results obtained.

POP[7] Recommends the top most popular items in all sessions.

SPOP[7] Recommends the top most popular items in the current session followed by the most popular ones in all sessions.

MARKOV-1[7, 9] Markov first order recommender. For each item it_j (first-order) it gathers which items and how many times happened to succeed after it_j inside all sessions.

Method	R-PREC	Cut-off @20						
		MAP	MRR†	PREC	REC	HR	POP	COV
RSC15-64 dataset								
RND	0.0004	0.0004	0.0011	0.0003	0.0026	0.0068	0.0107	**1.0000**
POP	0.0145	0.0220	0.0477	0.0095	0.0815	0.1467	**0.5172**	0.00267
SPOP	0.1880	0.2082	0.3655	0.0311	0.3057	0.4529	0.4962	0.5156
MARKOV-1	0.1184	0.2621	0.4862	**0.0641**	0.5472	0.7212	0.1290	0.7639
SKNN	0.2275	0.2810	0.4893	0.0576	0.5300	0.7258	0.1767	0.9127
VSKNN	**0.2355**	**0.2948**	**0.5117**	0.0628	**0.5721**	**0.7639**	0.1271	0.9471
RR-5 dataset								
RND	0.0000	0.0001	0.0002	0.0001	0.0005	0.0011	0.0134	**1.0000**
POP	0.0012	0.0017	0.0040	0.0010	0.0086	0.0186	**0.4887**	0.0005
SPOP	**0.3477**	0.3572	0.6118	0.0367	0.3999	0.6412	0.4594	0.4175
MARKOV-1	0.1211	0.2871	0.4989	**0.0574**	0.5350	0.6355	0.0423	0.6279
SKNN	0.3371	0.3800	0.6436	0.0540	**0.5490**	**0.7937**	0.0481	0.8665
VSKNN	0.3458	**0.3863**	**0.6579**	0.0535	0.5446	0.7901	0.0381	0.8962

Table 4.4: Results of nearest neighbours and frequency-based approaches. † - We used a non-standard MRR@k variant which we explained before in section 4.2.1.4. **Bold** - Best result. Underline - Second best result

Recommendation is based on these learned most popular state transitions (previous item → next item).

SKNN[7–9] A session k nearest neighbors algorithm based on the co-occurrence of items in sessions.

VSKNN[7–9] An improvement of SKNN, where now the temporal dimension is also taken in account through recency, inserted through the computations of the session similarity and the final item ranking scores. This is explained in detail on section 3.6

It is clear for both datasets that VSKNN is the overall winner method, with an advantage in the three main performance metrics, R-PREC, MAP, and MRR, while also providing more diverse recommendations for both datasets. This method is mostly followed by SKNN, which shows that the temporal characteristics that VSKNN takes into account on top of SKNN are meaningful and that taking into account the temporal dimension through recency has relevance in the session-based recommendation setting. It also shows that recency also helps in recommending less popular items. VSKNN is also the clear winner in providing recommendations with at least one relevant item evidenced by the results obtained for HR@20.

We can also observe the importance of item reminding for RR-5. When a user interacts with a particular item, he will likely interact with that item later on in the same session. This is explained by the surprising results obtained by SPOP for this dataset and go in hand with the findings of Twardowski, Zawistowski, and Zaborowski [7] that have shown

this phenomen to be more pronounced for RR-5 than RSC15-64. On the other hand, popularity has little importance for this dataset, as we can easily observe a significant difference in the POP results for the top two most performant methods, VSKNN and SPOP. Latifi, Mauro, and Jannach [2] calls this strategy REMIND, where a model reminds a user about a previously interacted item. They have shown that the most performant model in their testbed for RR-5 is a combination of VSKNN with a REMIND strategy. SPOP and VSKNN obtain very close results in the main performance metrics for this dataset, but their predictions are very different in item diversity, where VSKNN is much superior. VSKNN recommends more unpopular (POP) items in an overall more diverse spectrum (COV). Besides this, the HR@20 signals that SPOP produces more recommendations without a single relevant item than VSKNN for this case.

Session-based k Nearest Neighbors (SKNN and VSKNN) have a significant weakness regarding their inability to produce a ranked list for all items in the dataset in specific scenarios. Both these methods start by limiting their neighborhood to the sessions that have at least one item in common with our current session, and then we limit the scope of the items to recommend to those items inside those matched sessions. There are cases where the sessions included in such a neighborhood have less than k different items together. The list will always have less than the k requested events, and it can even be an empty list in cold-start scenarios.

4.5 Cross vs. Joint Deep Metric Learning Encoder Architectures

This section presents the experiments made on the metric learning methods of different variants related to the item encoder and item embedding layer. This includes a total of three variants, Twardowski, Zawistowski, and Zaborowski [7] originally proposed the first two DML-Cross-* models, and the third model is our proposed DML-Joint-*. We also experiment with two different loss functions: the classical Triplet-Loss and the SDML loss function.

DML-Cross-Full. Model proposed by Twardowski, Zawistowski, and Zaborowski [7] as an initial attempt to apply metric learning models for session-based recommendation systems. Sessions and Items encoding is separate in their exclusive encoders. It has two separate but equal in shape item embedding layers, one for the item encoder and another for the session encoder. The item encoder will incorporate the item's global representations, while the layer on the session encoder encompasses the item's intra-session representation. As we explained before in detail in section 3.3.1, we believe this separation does not lead to a relevant gain, and it just be redundant.

DML-Cross-Shared. An improvement also proposed on the same work [7] from the previous model, where the item embedding layer is now a layer shared between both

Method	R-PREC	Cut-off @20						
		MAP	MRR†	PREC	REC	HR	POP	COV
RSC15-64 dataset								
DML-Cross-Full-Triplet	0.1809	0.2378	0.4195	0.0577	0.5295	0.7169	0.1279	**0.9584**
DML-Cross-Full-SDML	0.2165	0.2755	0.4831	0.0619	0.5637	0.7478	0.0877	0.7956
DML-Cross-Shared-Triplet	0.2041	0.2599	0.4539	0.0595	0.5455	0.7348	0.1294	0.9528
DML-Cross-Shared-SDML	0.2208	0.2807	0.4909	0.0624	0.5697	0.7550	0.0903	0.7498
DML-Joint-Triplet	**0.2237**	**0.2845**	**0.4954**	**0.0630**	**0.5746**	**0.7642**	**0.1364**	0.9168
DML-Joint-SDML	0.2218	0.2813	0.4913	0.0627	0.5716	0.7585	0.0896	0.7854
RR-5 dataset								
DML-Cross-Full-Triplet	0.1563	0.1859	0.3133	0.0330	0.3504	0.5433	0.0460	**0.9560**
DML-Cross-Full-SDML	0.2970	0.3350	0.5696	0.0509	0.5181	0.7528	0.0486	0.7358
DML-Cross-Shared-Triplet	0.2851	0.3146	0.5346	0.0447	0.4662	0.7094	0.0444	0.9474
DML-Cross-Shared-SDML	0.1940	0.2286	0.3933	0.0413	0.4192	0.6263	**0.0533**	0.6606
DML-Joint-Triplet	**0.3283**	**0.3632**	**0.6182**	**0.0515**	**0.5266**	**0.7726**	0.0470	0.9054
DML-Joint-SDML	0.2306	0.2683	0.4625	0.0449	0.4571	0.6739	0.0517	0.6275

Table 4.5: Results for the DML models experimenting with 3 different loss functions and three different architectures relative to the item embedding layer. † - We used a non-standard MRR@k variant which we explained before in section 4.2.1.4. **Bold** - Best result. Underline - Second best result

item and session encoders. Their ablation study[4] concluded that this strategy led to higher results for Recall@20 and MRR@20 for their setup. Besides this gain in performance, previously in section 3.3.2 we formally analyzed the significant parameter reduction that this simplification entails.

DML-Joint. Our proposed model (sec. 3.3.2), where the item encoder is now only comprised of an item embedding layer, and this same layer is also shared with the sequence encoder. This model can be considered a further simplification of the previous model.

4.5.1 Results Analysis - Proposed Item Embedding Layer Perspective

Observing the results for RSC15-64 and RR-5 in table 4.5, it is clear that using a shared item embedding layer leads to superior results, confirming the claim made by Twardowski, Zawistowski, and Zaborowski [7]. We also notice that our proposed DML-Joint-* models further enhance the performance across all different loss functions except when using the SDML loss function for the RR-5 dataset. This improvement is consensual across all performance metrics in our experiments, either k-agnostic or not. We notice that the most performant solution with a shared embedding layer tends to recommend slightly more popular items (POP@20) with a smaller recommendation diversity overall

[4]Using RNN and MaxPool as the sequence encoders and Triplet-Loss as the loss function

(COV@20), a strategy that seems to pay off when we look at the main results. A significant advantage of our DML-Joint model is the increase in HR@20, showing that it results in fewer recommendations in which there is no single relevant item in the recommendation provided.

In summary, our proposed contribution relative to the item encoding technique leads to an increase in model performance across all metrics.

4.5.2 Results Analysis - Loss Functions Perspective

As we can see in table 4.5 the results are clarifying and make it possible to draw some interesting conclusions regarding the loss functions.

Although SDML is undoubtedly the best choice for the more complex DML-Cross-Full-* models, it does not progress in performance when we resort to simpler item encoding techniques. This loss function when combined with simpler item encoding techniques even led to a significant decrease in performance for the models trained in RR-5.

We believe the inherent regularization effect induced by SDML (sec. 3.5.1.1) is working to attenuate any overfitting that results from the more complex models, but in contrast, it limits the performance of SDML against simpler models. Contrary to SDML, we notice that triplet-loss can take advantage of the incremental simplification of the item encoding and the best result actually arises from combining our proposed joint encoding technique with triplet-loss.

4.6 Content-Based Learning

In this section, we analyze the consequences of combining collaborative filtering and content-based learning for metric learning based on our two different proposed variants: Item2Discrete (sec. 3.4.1) and Item2Vec (sec. 3.4.2). Given the results obtained in the previous section (4.5) relative to the item encoding, we now focus our experiments on top of the DML-Joint model.

As can be seen from table 4.6, our attempts to combine content-based learning with collaborative filtering only seem to improve the models that use SDML as the loss function. In the case of triplet-loss, we observe a decrease in model performance for both datasets.

Doing a more careful inspection, despite this negative tendency, we observe that for the models that use SDML as the loss function with Item2Vec, we can verify a slight improvement shown in all performance metrics for both datasets, but not enough to surpass the collaborative-filtering only DML-Joint-Triplet models, where the gap is quite significant for the RR-5 dataset. One interesting finding is that incorporating these content-based learning techniques makes the model resort more to popular recommendations but at the cost of worse performance and more recommendations that comprise the worst scenario, where there is no single relevant prediction at the top-20.

Method	R-PREC	Cut-off @20						
		MAP	MRR†	PREC	REC	HR	POP	COV
RSC15-64 dataset								
DML-Joint-Triplet	**0.2237**	**0.2845**	**0.4954**	**0.0630**	**0.5746**	**0.7642**	0.1364	0.9168
DML-Joint-Triplet-Item2Discrete	0.2178	0.2769	0.4825	0.0620	0.5657	0.7549	0.1370	**0.9250**
DML-Joint-Triplet-Item2Vec	0.2118	0.2684	0.4688	0.0606	0.5530	0.7440	**0.1392**	0.9142
DML-Joint-SDML	0.2218	0.2813	0.4913	0.0627	0.5716	0.7585	0.0896	0.7854
DML-Joint-SDML-Item2Discrete	0.2209	0.2803	0.4900	0.0628	0.5727	0.7591	0.0901	0.7833
DML-Joint-SDML-Item2Vec	0.2223	0.2819	0.4925	0.0629	0.5737	0.7612	0.0893	0.7846
RR-5 dataset								
DML-Joint-Triplet	**0.3283**	**0.3632**	**0.6182**	**0.0515**	**0.5266**	**0.7726**	0.0470	0.9054
DML-Joint-Triplet-Item2Discrete	0.3170	0.3500	0.5969	0.0499	0.5119	0.7586	0.0477	**0.9066**
DML-Joint-Triplet-Item2Vec	0.3175	0.3511	0.5986	0.0500	0.5127	0.7576	0.0481	0.9025
DML-Joint-SDML	0.2306	0.2683	0.4625	0.0449	0.4571	0.6739	0.0517	0.6275
DML-Joint-SDML-Item2Discrete	0.2316	0.2698	0.4685	0.0457	0.4633	0.6837	0.0534	0.5193
DML-Joint-SDML-Item2Vec	0.2391	0.2771	0.4792	0.0462	0.4689	0.6908	**0.0542**	0.5262

Table 4.6: Results with collaborative filtering only versus models that include the proposed content-based learning techniques Item2Discrete and Item2Vec. † - We used a non-standard MRR@k variant which we explained before in section 4.2.1.4. **Bold** - Best result. Underline - Second best result

In section 3.4 we explained that both techniques train using the available event and item metadata, and learn an initial embedding representation of the items, which is then the initial starting point in the collaborative-filtering DML models. Given this, we believe SDML benefits from using these content-based learning techniques models because its inherent regularization effect shown to provide more stable results can take advantage of this embedded information while at the same time containing the noisiness induced by this technique.

We believe the datasets choice have a significant impact on the results obtained. RSC15-64 only has the *categoryid* and event type properties, and most of the properties available on RR-5 are noisy, so we were only able to confidently use 2 item properties combined with the event type as our content metadata. More importantly, we are severely limited in maneuvering such metadata, given they are hashed for privacy reasons.

In summary, we can conclude that our Item2Vec proposed content-based learning techniques are helpful if combined with a more stable loss function like SDML, but harmful for more strict and sensitive models with triplet-loss, besides adding a slight bias for a more popularity-based recommendation.

4.7 Temporal-assisted approaches

This section focuses on experiments conducted in the models that incorporate temporal characteristics. The first approach, DML-Joint-TripletW, does this through the loss function, while our second approach, DML-Joint-TripletW-TRR, uses the robust VSKNN as a temporal re-ranker for the DML model, as detailed in section 3.6.2.

Method	R-PREC	Cut-off @20						
		MAP	MRR†	PREC	REC	HR	POP	COV
RSC15-64 dataset								
DML-Joint-Triplet	0.2237	0.2845	0.4954	0.0630	**0.5746**	**0.7642**	0.1364	0.9168
DML-Joint-TripletW	**0.2255**	**0.2849**	**0.4969**	**0.0631**	0.5741	0.7637	**0.1366**	0.8977
RR-5 dataset								
DML-Joint-Triplet	0.3283	0.3632	0.6182	0.0515	0.5266	0.7726	0.0470	0.9054
DML-Joint-TripletW	**0.3299**	**0.3639**	**0.6200**	**0.0516**	**0.5271**	**0.7745**	0.0468	0.9070

Table 4.7: Results for the DML models experimenting with two different triplet-loss variants, the original versus a modified weighted variant that incorporates temporal characteristics through recency in training. † - We used a non-standard MRR@k variant which we explained before in section 4.2.1.4. **Bold** - Best result. Underline - Second best result

4.7.1 Weighted Triplet-Loss

In section 3.5.1.2, we disclose a variant of the triplet-loss adapted to attribute different relevance to the positive samples/items based on their position in the ground-truth: a weighted triplet-loss. This loss function incorporates the temporal dimension in learning through the temporal proximity towards subsequent items in the ground-truth, resulting in a more flexible and intuitive learned metric entailing a better structure subspace for recommendation.

From the results of table 4.7.1 we observe a slight increase on the performance metrics in accordance to what Twardowski, Zawistowski, and Zaborowski [7] obtained in their ablation study[5]. These results also back the idea behind the invention of VSKNN, where intra-session order of the events or the changing of user preferences over time can be relevant factors for session-based recommendation.

4.7.2 Temporal Re-Ranking

After seeing the benefit from taking advantage of temporal characteristics in VSKNN and the weighted triplet-loss, we further explored a new way to use this for enhanced performance. The use of a weighted triplet-loss takes advantage of the temporal-proximity of the current session to its ground-truth while training, but now we use VSKNN on top of our DML-Joint-TripletW (best model so far) as a temporal re-ranker. We varied its re-ranking strength, where a re-ranking weight of 0 means no re-ranking while a re-ranking of 1 is the same as using the original VSKNN (more details in sec. 3.6.2). We tested for the following reasonable weights: 0.1, 0.25, 0.5, 0.75, and 0.9.

This contribution leads to a further increase in performance on top of the previous contributions (table 4.7.2), where we are rewarded with the best results when resorting to a relatively high re-ranking weight of 0.75. This re-ranking gain in performance is

[5] A DML-Cross-Shared model but with RNN or MaxPool as the sequence encoders

Method	Temporal Weight	R-PREC	Cut-off @20						
			MAP	MRR†	PREC	REC	HR	POP	COV
RSC15-64 dataset									
DML-Joint-TripletW-TRR	0.10	0.2292	0.2891	0.5029	0.0635	0.5784	0.7670	0.1362	0.9031
	0.25	0.2314	0.2928	0.5088	0.0639	0.5820	0.7718	0.1358	0.9200
	0.50	0.2387	0.2993	0.5189	0.0649	0.5894	0.7775	**0.1377**	0.9192
	0.75	**0.2388**	**0.3011**	**0.5214**	**0.0656**	**0.5952**	**0.7833**	0.1376	0.9365
	0.90	0.2373	0.2994	0.5186	0.0652	0.5923	0.7821	0.1371	**0.9397**
DML-Joint-SDML-TRR	0.10	0.2238	0.2840	0.4950	0.0631	0.5749	0.7611	0.0908	0.7996
	0.25	0.2292	0.2892	0.5025	0.0634	0.5774	0.7641	0.0916	0.8175
	0.50	0.2344	0.2950	0.5115	0.0640	0.5822	0.7696	0.0950	0.8277
	0.75	0.2366	0.2983	0.5165	0.0649	0.5888	0.7760	0.1024	0.86335
	0.90	0.2356	0.2977	0.5156	0.0649	0.5887	0.7762	0.1155	0.8777
RR-5 dataset									
DML-Joint-TripletW-TRR	0.10	0.3372	0.3697	0.6298	0.0516	0.5269	0.7743	0.0478	0.8994
	0.25	0.3452	0.3782	0.6455	0.0522	0.5318	0.7792	0.0480	0.8937
	0.50	0.3525	0.3880	0.6629	0.0537	0.5444	0.7920	0.0463	0.9145
	0.75	**0.3540**	**0.3908**	**0.6668**	0.0545	0.5519	0.7978	0.0465	0.9170
	0.90	0.3526	0.3907	**0.6668**	**0.0551**	**0.5566**	**0.8024**	0.0450	**0.9275**
DML-Joint-SDML-TRR	0.10	0.2619	0.3022	0.5196	0.0482	0.4902	0.7168	0.0524	0.6021
	0.25	0.2846	0.3228	0.5529	0.0492	0.4998	0.7303	0.0530	0.5909
	0.50	0.3270	0.3604	0.6140	0.0507	0.5139	0.7475	**0.0533**	0.6139
	0.75	0.3318	0.3668	0.6246	0.0518	0.5233	0.7559	0.0528	0.6589
	0.90	0.3337	0.3700	0.6301	0.0526	0.5301	0.7632	0.0519	0.7063

Table 4.8: Results of the proposed metric learning model (collaborative filtering only) with temporal re-ranking from VSKNN. † - We used a non-standard MRR@k variant which we explained before in section 4.2.1.4. **Bold** - Best result. Underline - Second best result

more significant in the models applied on RR-5 that use the SDML loss function, where VSKNN re-ranking dramatically compensates for the lower performance of these models compared to those using the weighted Triplet-loss. Since we use VSKNN, not all the increase in performance comes from the recency factor that VSKNN takes into account and from its session k nearest neighbors approach. With this last proposed model, we attain our best model, the DML-Joint-TripletW-TRR, with a temporal re-ranking weight of 0.75.

4.8 DML Convergence Analysis

In order to assess the models training behaviour, we monitored the evolution of R-precision during training, on the validation set. Figure 4.4 shows the R-Precision evolution, for both the DML-Joint-TripletW (plots at the left) and DML-Joint-TripletW-TRR (plots on the right) models, on the two datasets.

The models are trained up to 100 epochs using a learning rate scheduler that decreases its rate when evaluation results on the primary metric, in our case R-Precision, on the validation set, decreases. By observing Figure 4.4, we can observe a steady increase in our primary metric R-Precision for our top 2 best models on both datasets, evidencing model

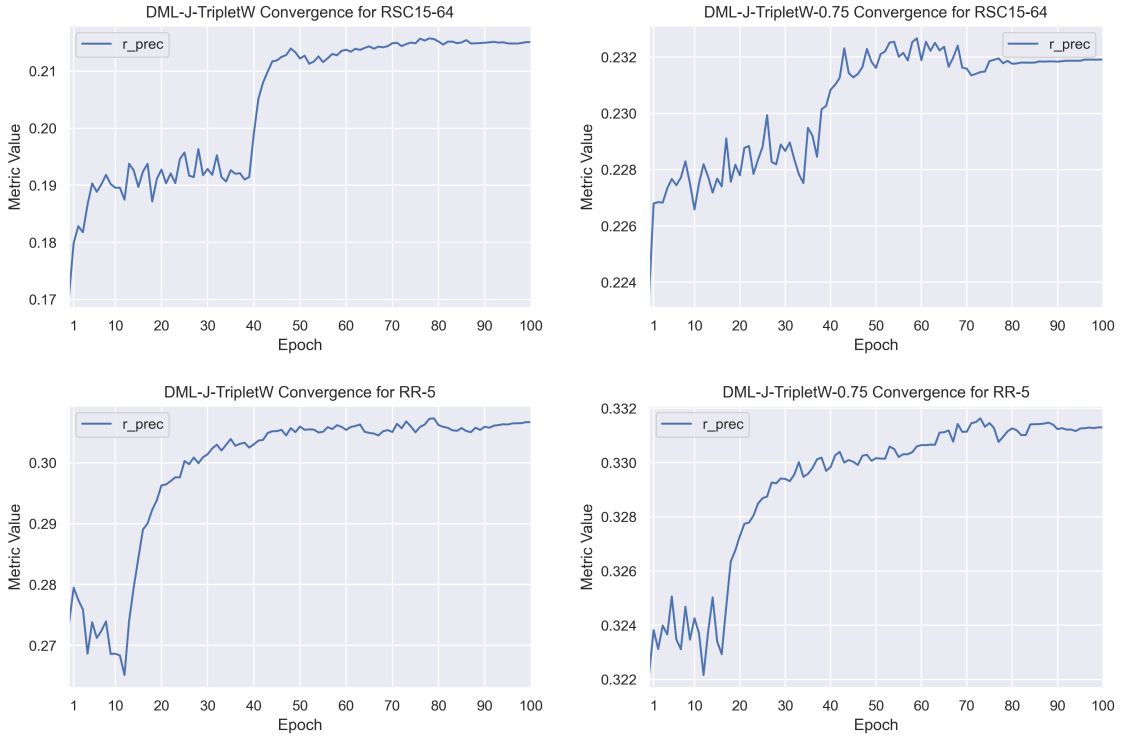


Figure 4.4: Convergence analysis for the metric learning models. On the left we have our metric learning models with a joint session-item encoding contribution, while on the right we have the model that performs Temporal Re-Ranking. At the top we have the models applied for RSC15-64 while at the bottom the models applied on the RR-5 dataset.

convergence. We can also observe that training for 100 epochs is enough to achieve the best performance, as performance stagnates in later epochs.

To complement this analysis, we conduct on Figure 4.5 the same analysis, now for the DML-Joint-TripletW model, but across all metrics. It can be seen that in general, performance increases as the training evolves, stabilizing after epoch 60.

4.9 Comparison with State-of-the-art

In this section, we summarize our main contributions and give awareness to the progress made in the field compared to the state-of-the-art DML-Cross-Shared-TripletW [7] and against the robust session k nearest neighbors technique, VSKNN. Given the unsatisfying results from our proposed techniques to use content-based learning in the learning process (sec. 4.6), such experiments are not discussed in this section.

In table 4.9 we see that the combination of our joint encoding technique, proposed in section 3.3.2, combined with a weighted triplet-loss that incorporates temporal-proximity,

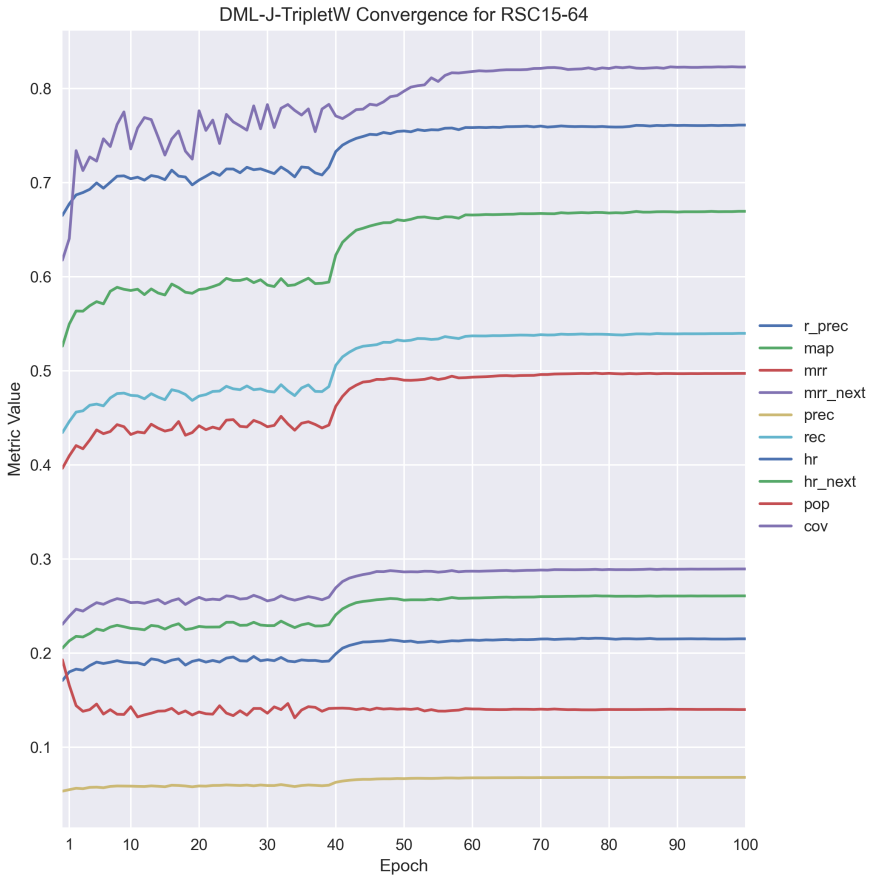


Figure 4.5: Convergence analysis, with all metrics, conducted on the proposed DML-Joint-TripletW model. All metrics are @20 except for R-Precision (r_prec on the figure) and both hr_next and mrr_next are respectively, HR@20 and MRR@20 for next-item prediction. (only the next item in the ground is the actual ground-truth)

results in a significant improvement compared to the state-of-the-art DML-Cross-Shared-TripletW.

Our proposed DML-Joint-TripletW model is further improved when one more contribution from this dissertation is added on top of it, the use of VSKNN as a temporal re-ranker for DML. This last model with temporal re-ranking, DML-Joint-TripletW-TRR, extends further the gap towards our starting point, the DML-Cross-Shared-TripletW, and even surpasses VSKNN itself for both datasets.

In summary, these results validate the relevance of most of our contributions for the field of session-based recommendation using metric-learning.

4.10 Understanding the Impact of Sessions Length

Session length analysis can be divided into two dimensions: (i) the analysis of varying input session length, and; (ii) the analysis of varying sizes of ground truth. The first

Method	R-PREC	Cut-off @20						
		MAP	MRR†	PREC	REC	HR	POP	COV
RSC15-64 dataset								
VSKNN	0.2355	0.2948	0.5117	0.0628	0.5721	0.7639	0.1271	**0.9471**
DML-Cross-Shared-TripletW	0.1979	0.2542	0.4475	0.0595	0.5444	0.7347	0.1278	0.9458
DML-Joint-TripletW	0.2255	0.2849	0.4969	0.0631	0.5741	0.7637	0.1366	0.8977
DML-Joint-TripletW-TRR ‡	**0.2388**	**0.3011**	**0.5214**	**0.0656**	**0.5952**	**0.7833**	**0.1376**	0.9365
RR-5 dataset								
VSKNN	0.3458	0.3863	0.6579	0.0535	0.5446	0.7901	0.0381	0.8962
DML-Cross-Shared-TripletW	0.2897	0.3186	0.5411	0.0450	0.4693	0.7117	0.0447	**0.9478**
DML-Joint-TripletW	0.3299	0.3639	0.6200	0.0516	0.5271	0.7745	**0.0468**	0.9070
DML-Joint-TripletW-TRR ‡	**0.3540**	**0.3908**	**0.6668**	**0.0545**	**0.5519**	**0.7978**	0.0465	0.9170

Table 4.9: Results comparing the best models as a combination of our several contributions against the state-of-the-art in this field, DML-Cross-Shared-TripletW of [7], and also against the robust session k nearest neighbors algorithm VSKNN of [9]. † - We used a non-standard MRR@k variant which we explained before in section 4.2.1.4. ‡ - Using a temporal re-ranking weight of 0.75. **Bold** - Best result. Underline - Second best result

approach observes how the models perform in the face of more or fewer input data and earlier or longer steps within a session. The second approach tries to obtain data with higher granularity related to the varying lengths of ground-truth. We analyze the impact of the varying session and ground truth lengths on the results. To clarify, we refer to the number of known past interactions within the same session as the input session length while the true hidden events that come after as our ground-truth also vary.

To analyze this, we have focused on the following metrics: R-PREC as the main metric to evaluate the overall performance of the prediction without requiring a specific cut-off; MAP@20, given it is the most common metric used in this field, the standard on the TREC EVAL community, with a primary focus on the overall ranked prediction list; HR@1 (adapted for next item prediction) to evaluate how good is the model for the challenging task of precisely predicting the next item the user interacts with; and finally HR@20 and HR@1 where the former sees if the model can at least predict one item in the top-20 items in the prediction list that is in the ground-truth, while the latter evaluates if the top item in the prediction list is on the ground-truth, which is an interesting scenario in case we can only recommend at most one item.

4.10.1 Input Session Length

Different input session lengths describe sessions at different steps in time. Ideally, a recommendation occurs when at least one interaction/event is provided as input. After each interaction, the system must provide its first recommendation input. The cycle continues after each interaction, where after each step, we dispose of more past data to make the next recommendation.

Our intuition leads us to believe that, in general, the metric-learning models (DMLs)

can better represent the session in the learned d-dimensional space by having more data related to the current session. We hypothesize that the models should benefit from higher input session lengths that provide more past data within the same session than shorter input session lengths. The same hypothesis can be said for more superficial session k nearest neighbors algorithms like SKNN or VSKNN since they base their search on item co-occurrence in sessions, so more extended sessions as input should allow for broader and more stable results.

Surprisingly, our hypothesis does not hold. As we can see on figure 4.10.1, all models are significantly biased to perform better recommendations for shorter session lengths. In overall, there is a decay of performance for longer sessions. This is contrary to our intuition but has a rather simple explanation, where the main culprit for this behavior is the long-tail distribution we observed in both datasets for either the frequency per session length and the frequency per item in section 4.1.2. The positive aspect of this is that the models are quite accurate in the first stages of the recommendation.

VSKNN resorts to techniques that hide its significant computational complexity in inference by only considering subsets of sessions used for the final predicted item ranking list. We notice that this weakness is reflected in the results for RSC15-64, because VSKKN's heuristic to select its initial candidate neighbor sessions is based on the heuristic that selects sessions where there is one item in common with our current input session. A longer input session inherently has a higher probability of VSKNN's initial search of candidate neighbors being broader and a broader spectrum. However, given that such broad search space is filtered according to recency and co-occurrence-based heuristics to control the significant computational complexity of such operation that ends up discarding part of these neighborhoods, this might negatively affect VSKNN. Conversely, from figure 4.10.1, we can see that this does not happen for RR-5 when we compare VSKNN to DML-Joint-TripletW that does not suffer from this. This may be a result of more pronounced item repetition in this session, as shown by [7] inside sessions as a counterweight to this.

This analysis was crucial to dissect the previous metric results with more detail, in this case across different session lengths. We verify that the dataset distribution relative to the session length dramatically influences the results obtained. The results are greatly influenced by the evaluation results performed for shorter session lengths as they dominate most of the samples used.

4.10.2 Ground Truth Session Length

In this section, similar to the previous one, we try to see how having different lengths of ground truth affects our overall results and how different models deal with this.

Looking at figures 4.8, 4.9 and 4.10.2 we see that VSKNN, DML-Joint-TripletW and DML-Joint-TripletW-TRR have very similar results with no apparent distinctive behavior between them. A clear difference is noticed when we compare the performance of the

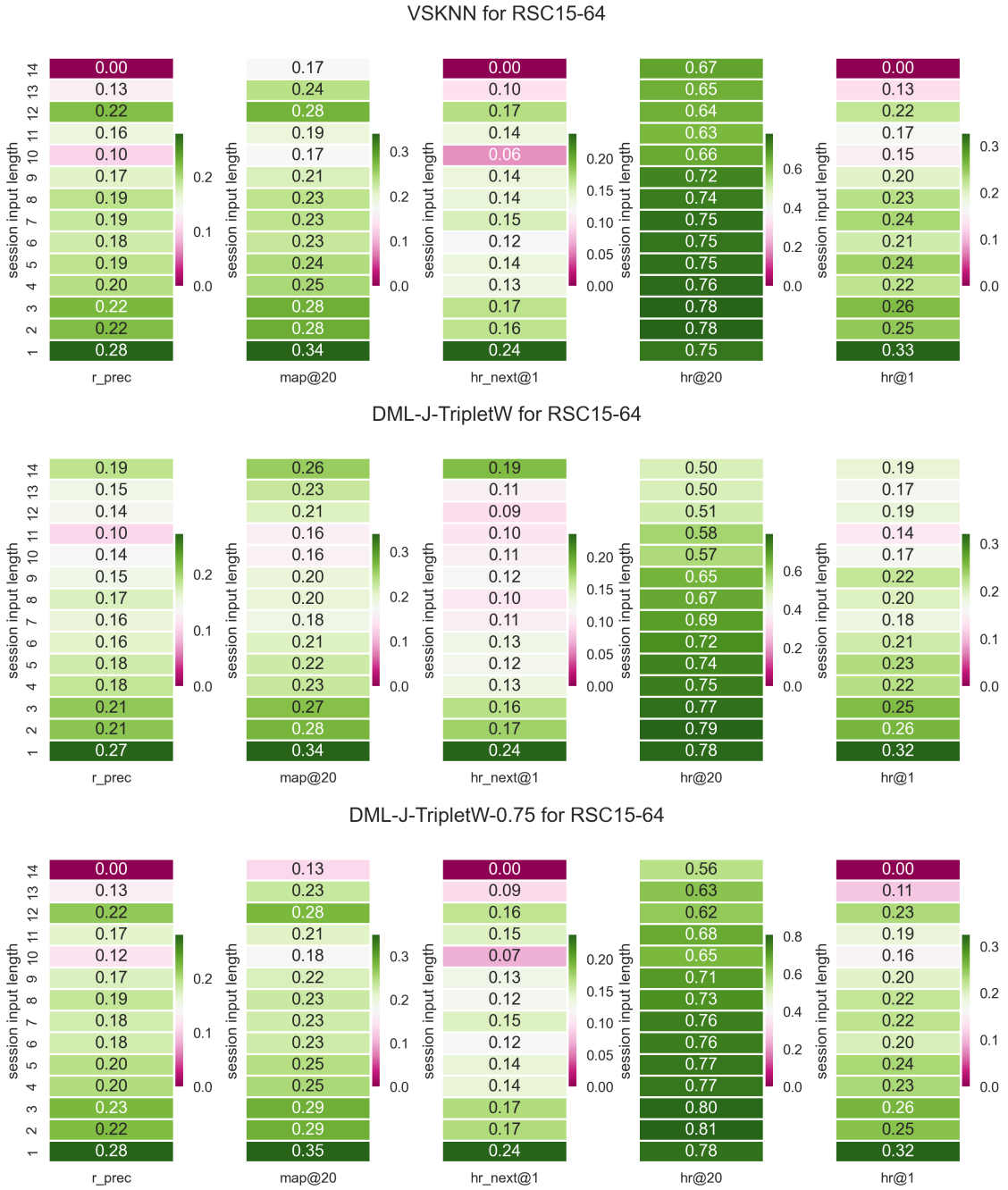


Figure 4.6: Average Metric Results evaluated on the test set per input session lengths.

models between RSC15-64 and RR-5, where for RSC15-64 the overall R-Precision and MAP@20 are much more stable across different ground truth lengths but that is not the same for RR-5. Making predictions with shorter ground-truth lengths is more challenging as it makes it more probable to have more misses in that prediction. We believe the significant higher performance of these models against RR-5 is not just due to the amount of dominant samples with a ground-truth length of one, but also due to the algorithms

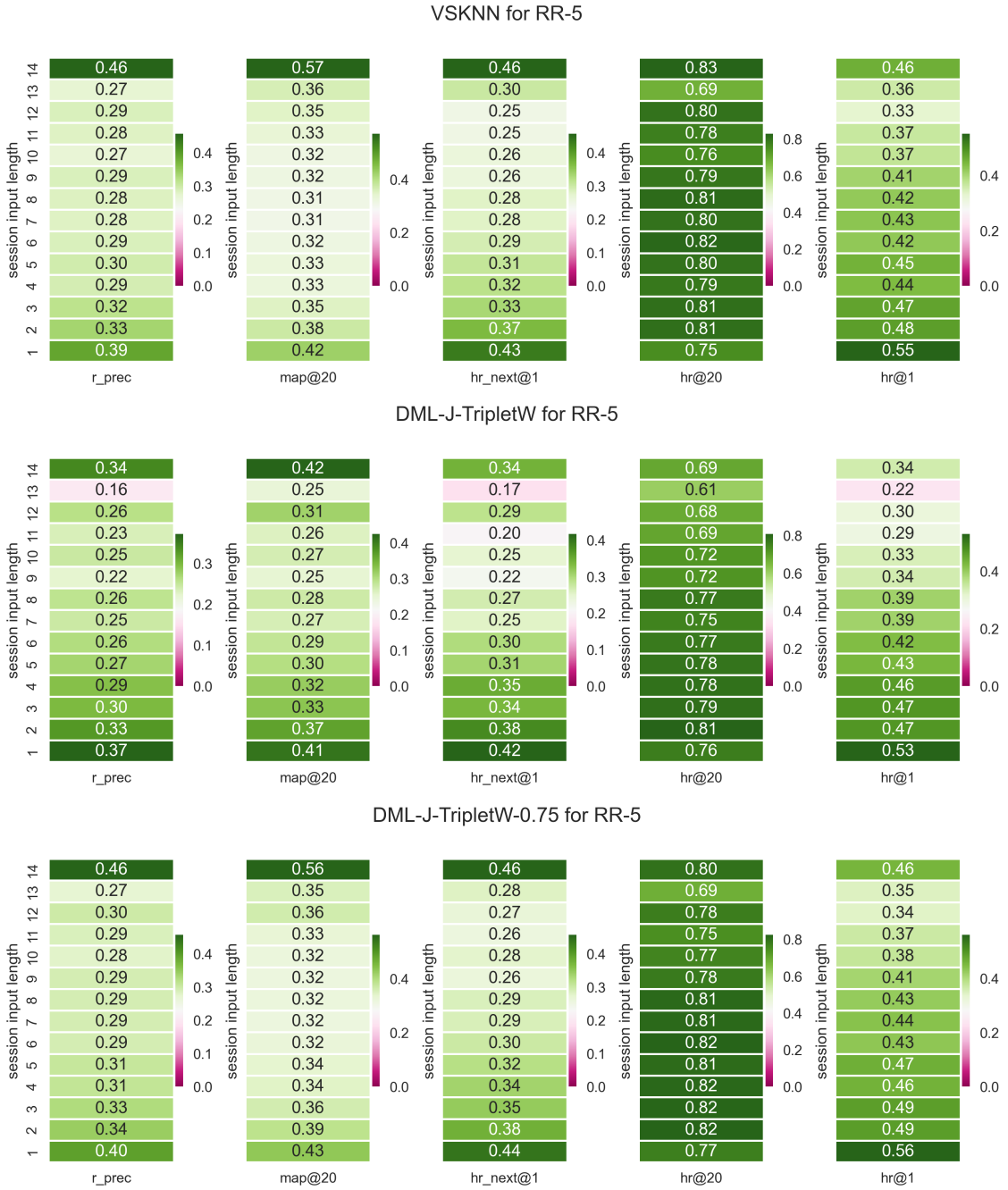


Figure 4.7: Average Metric Results evaluated on the test set per input session lengths.

benefiting from this more pronounced item repetition for RR-5. Since there is significantly more item repetition for RR-5, this first results in a reduced spectrum of possible candidate neighbor sessions and consequently reduced spectrum of items from those other sessions to recommend. If repetition of items tends to happen on those candidate sessions this is also accounted for in VSKNN, making its prediction task easier. On the other hand, DML might also be taking advantage of this through its CNN-based sequence

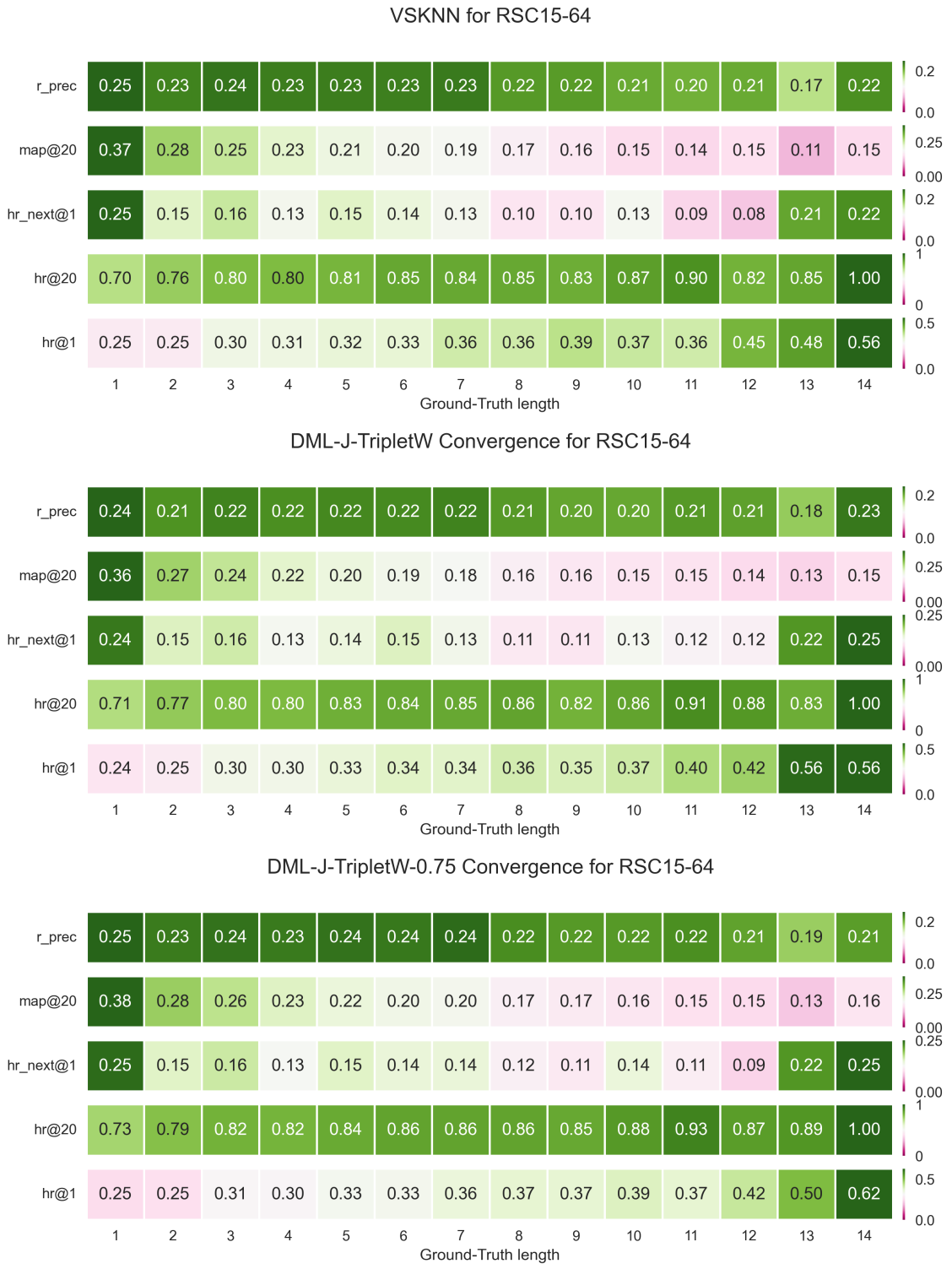


Figure 4.8: Average Metric Results evaluated on the test set per ground-truth length.

encoder that can detect spatial patterns inside sessions, and also from the metric-learned space that is also capable of learning dimensions that can represent such characteristic.

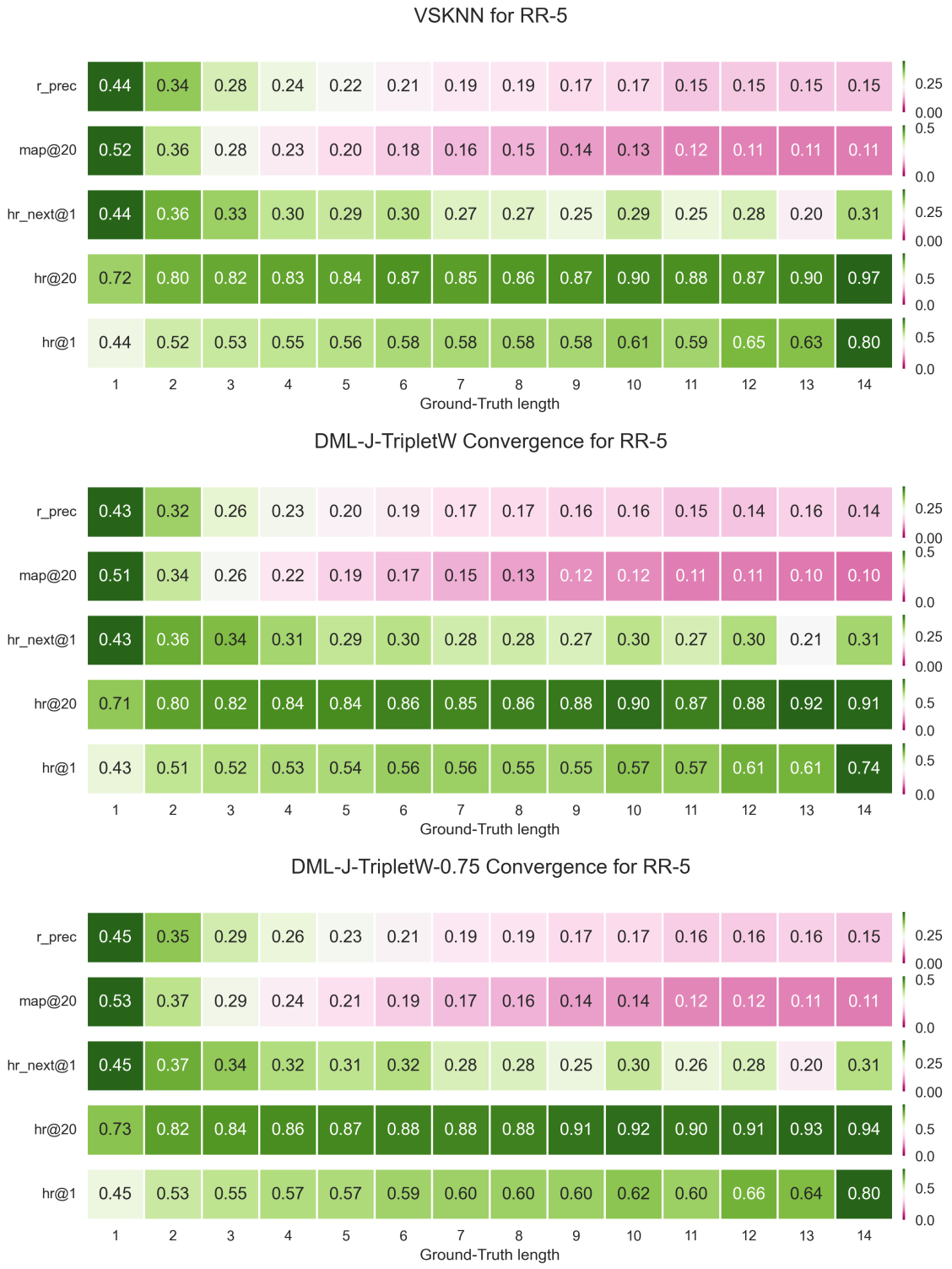


Figure 4.9: Average Metric Results evaluated on the test set per ground-truth length.

4.10.3 Input Session Length vs. Ground Truth Session Length

This section combines the input session length and the ground-truth length to see their coordinated impact on model performance from a renewed perspective.

Observing the results for HR@1 (next-item prediction) in figures 4.10 and 4.11, we can see how significantly easier is the task of next-item recommendation for the RR-5 dataset compared to RSC15-64 which we discussed previously. We can also see on figures 4.12 and 4.13 that VSKNN and DML-Joint-TripletW obtain similar results between them, but significant differences arise when we compared either model's results between RR-5 and RSC15-64.

These datasets were collected from websites with different characteristics and features, where interactions are certainly influenced by external processes not discriminated in either dataset (e.g., advertisement). This explains why using fundamentally different models, VSKNN and DML-Joint-TripletW lead to similar results when compared together, while at the same time, we observe a clear difference in results when running the same models on a different dataset. The datasets themselves play a more critical role in the results [10] than the choice of either model, and since certainly not all data relevant to each interaction is collected and used in training, we can consider this a limiting factor in attaining better results.

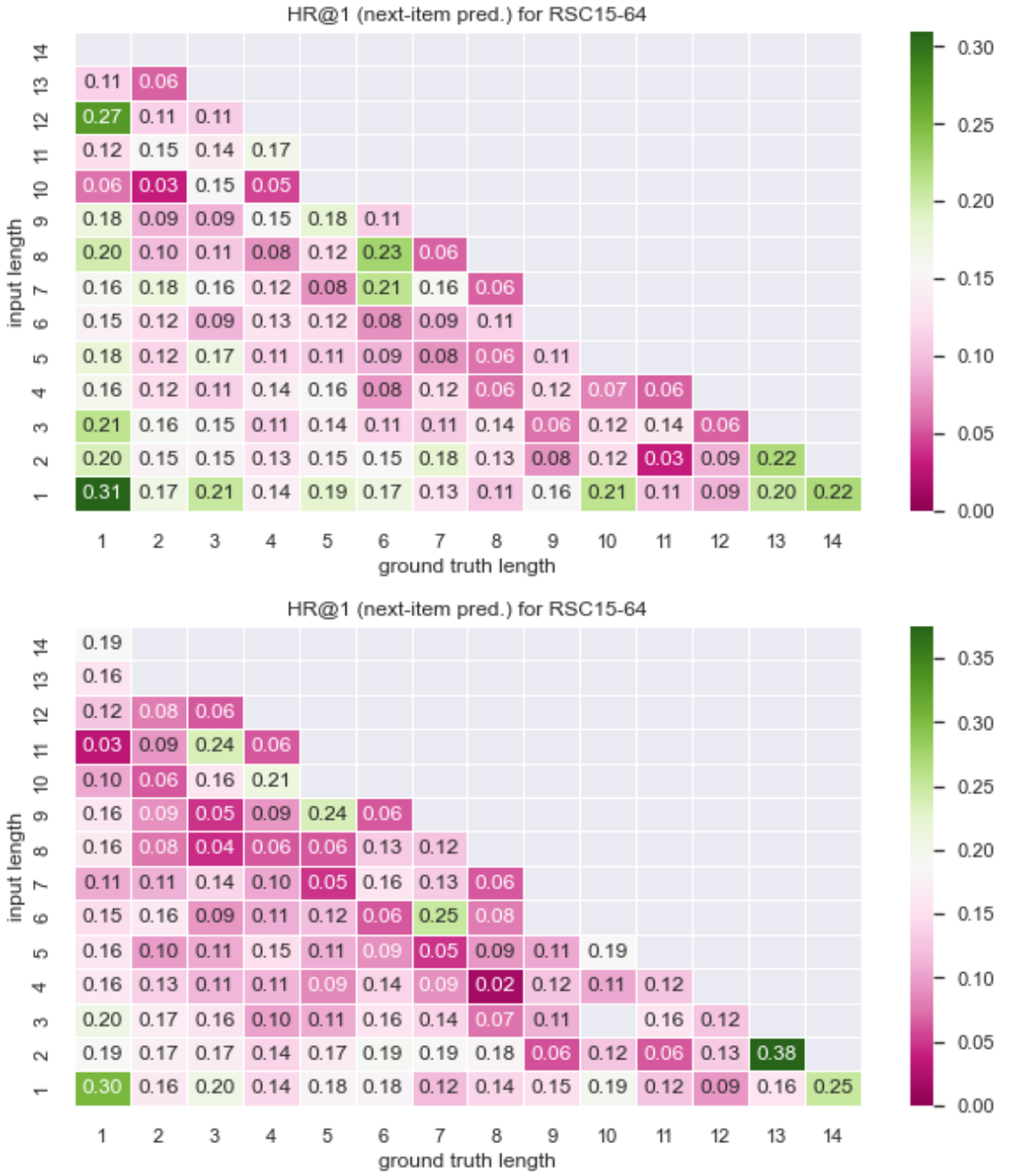


Figure 4.10: Results of HR@1 (next-item prediction) for VSKNN (top) and DML-Joint-TripletW (bottom) per input session length and ground-truth session length - RSC15-64 dataset

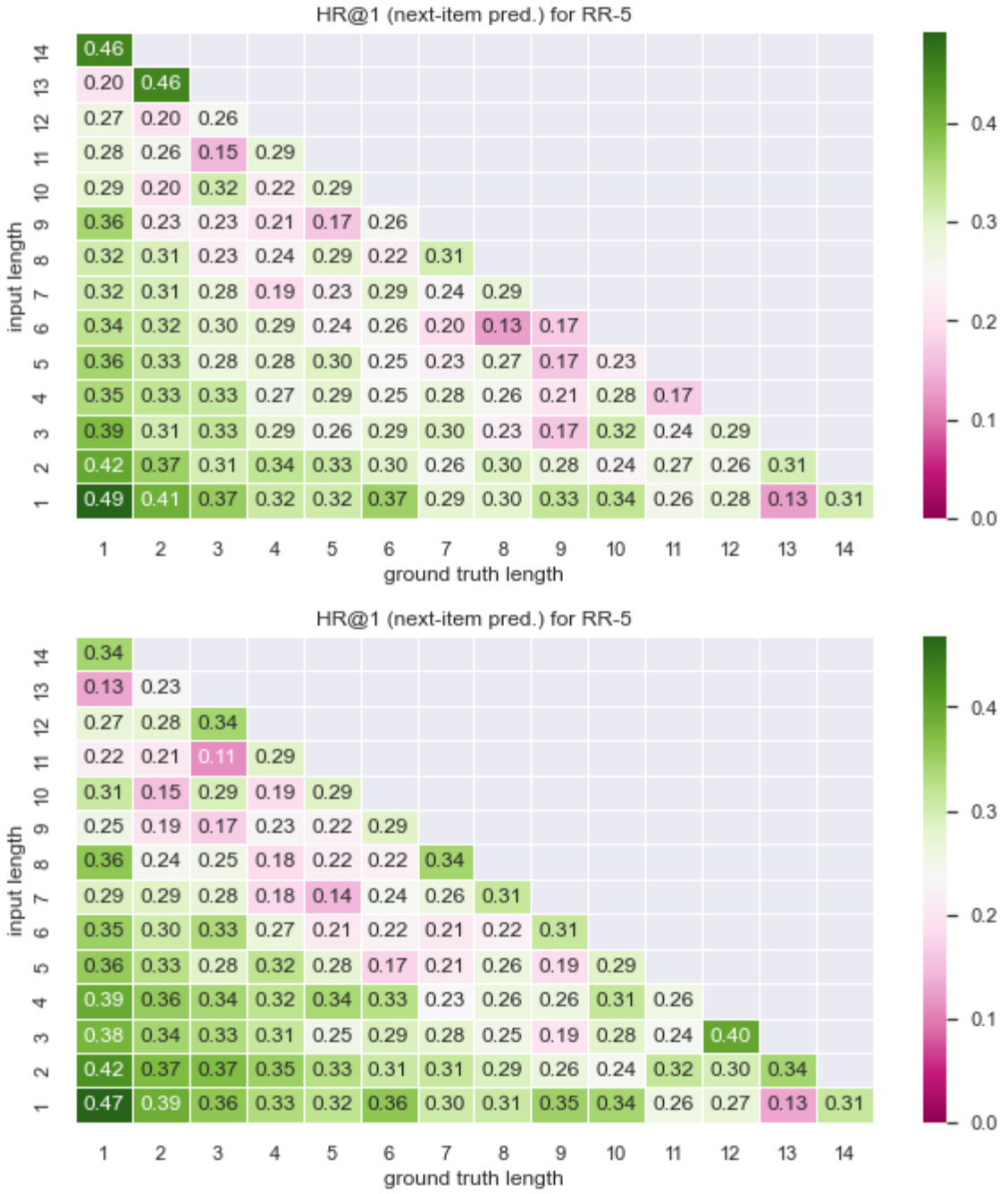


Figure 4.11: Results of HR@1 (next-item prediction) for VSKNN (top) and DML-Joint-TripletW (bottom) per input session length and ground-truth session length - RR-5 dataset

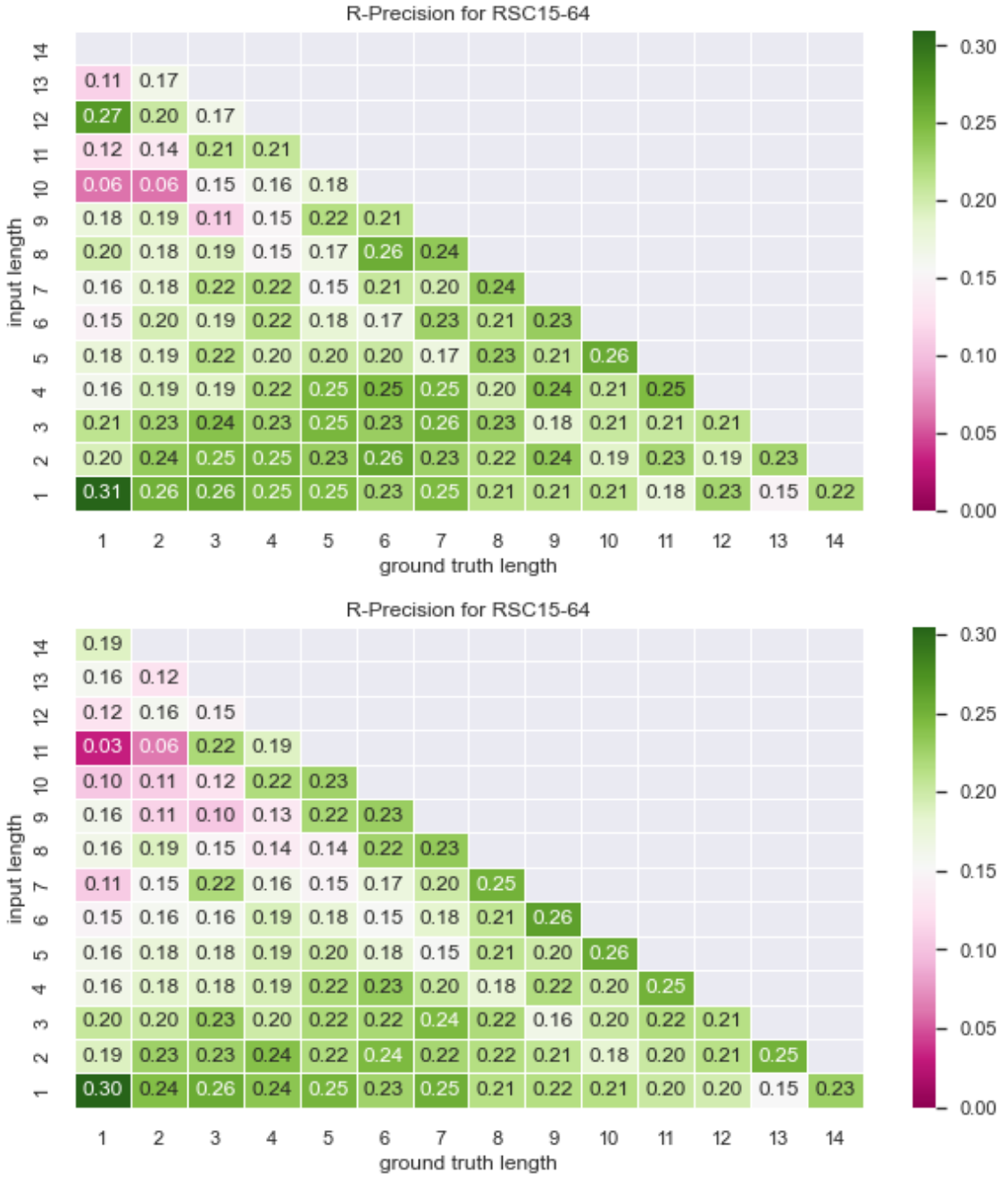


Figure 4.12: Results of R-Precision for VSKNN (top) and DML-Joint-TripletW (bottom) per input session length and ground-truth session length - RSC15-64 dataset

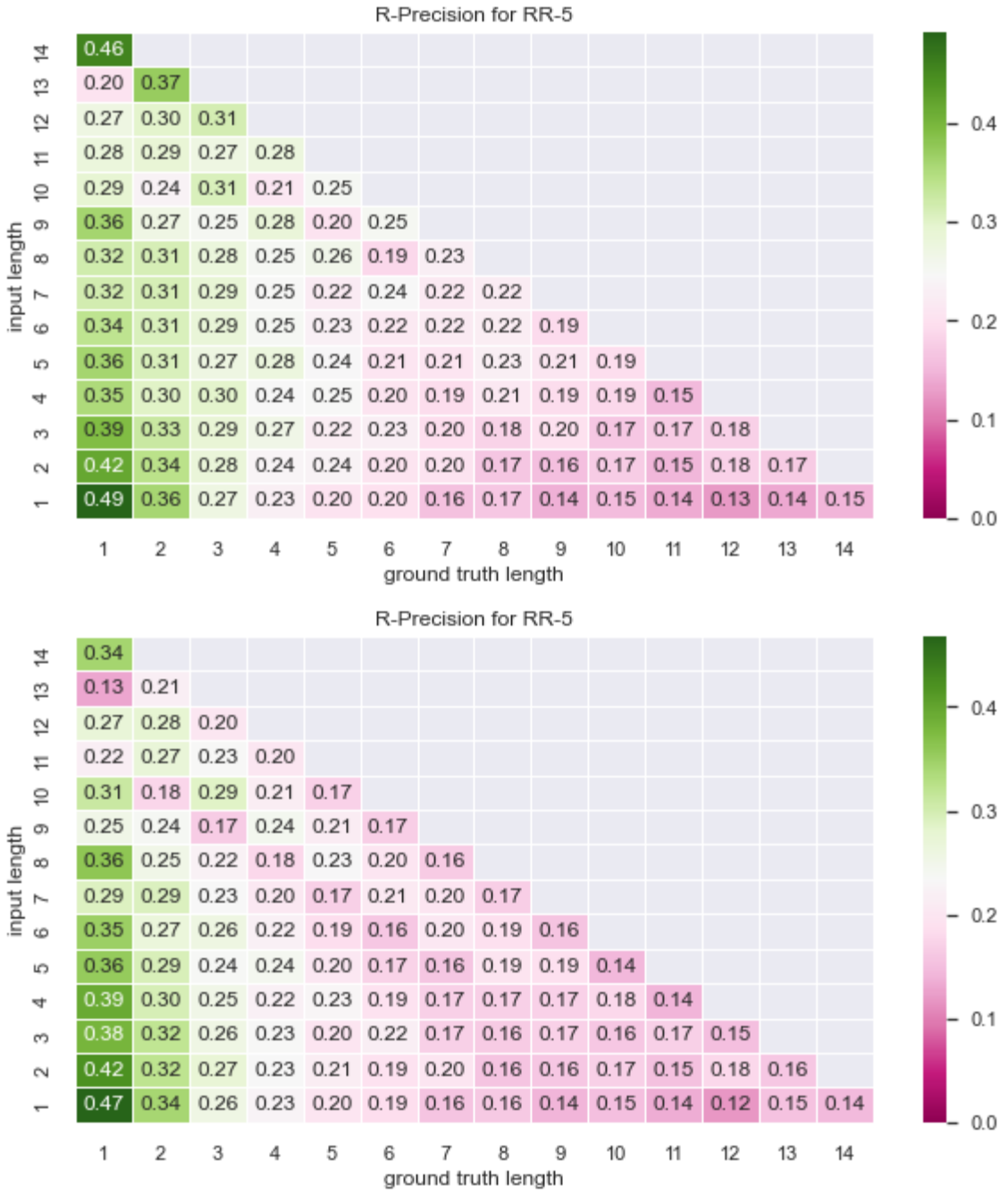


Figure 4.13: Results of R-Precision for VSKNN (top) and DML-Joint-TripletW (bottom) per input session length and ground-truth session length - RR-5 dataset

Method	R-Precision Gain (%)
DML-Joint-TripletW	0.00%
VSKNN	+4.42%
DML-Joint-TripletW-TRR	+6.19%

Table 4.10: Percentage gain on R-Precision compared to DML-Joint-TripletW averaged for both datasets.

4.11 Cost vs. Benefit Discussion

Previously from chapter 3.7 we discussed the different computational complexities involved specifically for inference. Below we summarize the computational complexity of our top two proposed models and VSKNN:

$$\begin{aligned} O(|I|) &\qquad \text{(DML-Joint-TripletW)} \\ O(|I|+|S|) &\qquad \text{(VSKNN or DML-Joint-TripletW-TRR)} \end{aligned} \tag{4.11}$$

Revising in table 4.9 the results compared across each of these models for both datasets, we have shown that at the top, we would have the DML-Joint-TripletW-TRR, followed by VSKNN, and finally by DML-Joint-TripletW. Table 4.11 illustrates the percentage increase of the R-Precision results averaged over both datasets.

Considering both dimensions, the computational costs (inference) in equation 4.11 and the performance gains in table 4.11 we can draw the following conclusion: the DML-Joint-TripletW-TRR and VSKNN are the most performant models but at a significant computational cost increase. At the same time, DML-Joint-TripletW provides fewer quality recommendations but at a significantly reduced computational cost. The usage of each model is dependent on the use case and the context it inserts. Below we gather and discuss a list of different use cases (also summarized in table 4.11), where each model is better suited for:

Best quality recommendations If the primary goal is to obtain the best recommendations, the choice falls under our proposed DML-Joint-TripletW-TRR.

Real-time recommendation In this context, recommendations are performed live, and the inference response time is of utmost importance to avoid a bad user experience. DML-Joint-TripletW is the winner for large-scale real-time recommendation, given its computational inference complexity at a slight decrease in recommendation quality compared to the other two methods.

Online training In this context, our goal is a model that can learn with new samples in something close to real time. VSKNN already computes everything from scratch at each inference request so essentially we are forced to perform online training with

Use Case (Priority)	Recommended Model
Best quality recommendations	DML-Joint-TripletW-TRR
Real-time recommendation	DML-Joint-TripletW
Online training	VSKNN
Data consistency	VSKNN

Table 4.11: Most adequate models per use cases.

it. On the other hand, for the DML models, as they are based on a neural network architecture, training such models in face of new samples is an infeasible task.

Data consistency As VSKNN essentially computes everything on the inference phase, it always takes advantage of the most up-to-date data, while for the neural network models, the models would need to be trained periodically and not at every inference round due to their high training and consequently financial cost to train.

5

Conclusions and Future Work

The use of metric-learning for session-based recommendation is a promising research field. In this dissertation, we proposed key novel ideas that improved over the state of the art [7]. The key proposed methods are:

- **(a) Novel joint session-item encoding model with temporal smoothing.** We propose a novel architecture targeting item encoding, simpler than [7], where session and item encoders are jointly-learned in a common space, serving as an intermediary for session-based recommendation. We show that this contribution produces noteworthy improvements while at the same time significantly reducing the number of parameters involved. We also propose two different content-based learning techniques to kick-start the collaborative-filtering metric-learning model, although this did not contributed to enhanced performance. Finally, we propose to leverage the temporal characteristics in two dimensions: temporal proximity and temporal recency.

- **(b) Outperform other state-of-the-art metric-learning models for session-based recommendation.** We outperform other[7] state-of-the-art metric-learning models for session-based recommendation. We are also able to outperform other quite robust session k nearest neighbors techniques like SKNN and VSKNN when taking advantage of the temporal features.

- **(c) Critical Analysis.** In this dissertation, we offer a critical analysis of the many dimensions of session-based recommendation in general. We analyze and bring awareness to certain topics we feel to be particularly important, like the computational complexity of the models as a relevant factor to consider besides the metric evaluation results, the in-depth analysis on the impact of dataset characteristics in the results, and a thorough discussion of the evaluation protocol for the recommendation domain. We also discuss several inconsistencies and mistakes commonly performed in this field related to the experimental protocol that creates significant confusion in the interpretation and comparison of similar research, a phenomenon called phantom progress[10, 11].

Use Case (Priority)	Recommended Model
Best quality recommendations	DML-Joint-TripletW-TRR
Real-time recommendation	DML-Joint-TripletW
Online training	VSKNN
Data consistency	VSKNN

Table 5.1: Most adequate models per use cases.

This dissertation also focused on researching the impact of dataset characteristics, namely how the session length and ground truth impact the results obtained. Our analysis confirms the significant impact of the long-tail distribution, a tendency in this field, on the results obtained. This imbalanced distribution is reflected in the models' performance, wherein the dominant shorter sessions are more effective and produce higher quality recommendations than longer sessions. The datasets openly available in this field are relatively limited in terms of metadata, particularly for the amount and quality of the metadata available. Other works [10], have also raised awareness on how the datasets have an impact on model performance. Nevertheless, we are already making steps towards addressing these issues.

We also provided a critical analysis of the experimental methodology followed in session-based recommendation.

Besides the promising results obtained, we conducted an asymptotic analysis, where we show that the model from contribution (a) surpasses other state-of-the-art models in the field. Even though it is not enough to surpass the robust VSKNN results, our proposed model surpasses VSKNN in terms of reduced computational complexity for inference. This computational complexity advantage allows our model to be the best deployment choice in more demanding instances than VSKNN. In table 5 we present a summary of where each model fits best, which is derived from a balanced consideration of the results obtained in our experiences and the asymptotic analysis conducted in this work.

In summary, contribution (a) results in a model that provides meaningful recommendations and is ideal for high-demanding inference scenarios. From the combination of contributions (a) and (c), we propose a further enhanced model more powerful than in (a), better suited for scenarios where live/online recommendation is not a requirement, but instead, the focus is on the quality of the recommendations. This last model even surpasses the robust VSKNN while maintaining on-par computational complexity.

5.1 Future Work

There are a number of possible next steps aiming at improving the results and showing the generalization of the achieved results. In pragmatic terms, we foresee the following steps as the most promising ones:

- **BBC video recommendation.** In collaboration with BBC, we have been jointly writing a paper[1] involving the findings of this dissertation applied on their proprietary BBC-VoD dataset, a real-world proprietary dataset containing daily interactions made by users in BBC's iPlayer streaming platform. This paper had the goal of corroborating our work against a real-world use case with raw and unfiltered data and provide a reliable solution for session-based recommendation in the iPlayer platform.

- **Transformer-based recommendation.** One interesting approach can be the usage of a transformer-based architecture using metric learning for session-based recommendation. Such models have proven incredibly powerful for a variety of tasks[18, 25, 59–63] and intuitively we can see that attention may play an important role for session-based recommendation. The simplest way to do this would be to replace our TagSpace sequence encoder with a transformer, while at the same time adapting the transformer itself for metric-learning.

- **Temporal dimension.** Incorporating the temporal proximity factor on the computations of the triplet-loss contributes positively to the model. Given this finding and from other works [41, 42], we are confident that the use of adaptive margins is a promising path to follow on top of our work. This technique has even been used as a means to incorporate temporal characteristics [42] in metric-learning.

[1] https://github.com/jgrodrigues/dml-joint

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. ii).

[2] S. Latifi, N. Mauro, and D. Jannach. "Session-aware recommendation: A surprising quest for the state-of-the-art". In: *Information Sciences* 573 (2021), pp. 291–315 (cit. on pp. xv, 11–13, 51, 53).

[3] J. F. Kelley. "An Iterative Design Methodology for User-Friendly Natural Language Office Information Applications". In: *ACM Trans. Inf. Syst.* 2.1 (Jan. 1984), pp. 26–41. ISSN: 1046-8188. DOI: 10.1145/357417.357420. URL: https://doi.org/10.1145/357417.357420 (cit. on pp. xv, 16).

[4] A. Saha, M. M. Khapra, and K. Sankaranarayanan. "Towards Building Large Scale Multimodal Domain-Aware Conversation Systems". In: *AAAI*. 2018 (cit. on pp. 1, 15).

[5] L. Liao et al. "Knowledge-aware Multimodal Dialogue Systems". In: *Proceedings of the 26th ACM international conference on Multimedia* (2018) (cit. on pp. 1, 15, 16).

[6] C. Cui et al. "User Attention-Guided Multimodal Dialog Systems". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. Paris, France: Association for Computing Machinery, 2019, pp. 445–454. ISBN: 9781450361729. DOI: 10.1145/3331184.3331226. URL: https://doi.org/10.1145/3331184.3331226 (cit. on pp. 1, 16).

[7] B. Twardowski, P. Zawistowski, and S. Zaborowski. *Metric Learning for Session-based Recommendations*. 2021. arXiv: 2101.02655 [cs.IR] (cit. on pp. 2, 4, 18, 19, 21, 23, 27–31, 36, 38, 39, 43, 44, 47–49, 51–54, 57, 59, 61, 62, 74).

[8] D. Jannach and M. Ludewig. "When recurrent neural networks meet the neighborhood for session-based recommendation". In: *Proceedings of the Eleventh ACM Conference ...* (2017) (cit. on pp. 5, 39, 52).

[9] M. Ludewig and D. Jannach. "Evaluation of session-based recommendation algorithms". In: *User Modeling and User-Adapted Interaction* 28.4-5 (2018), pp. 331–390 (cit. on pp. 5, 33, 38, 39, 51, 52, 61).

[10] M. Ludewig et al. "Empirical analysis of session-based recommendation algorithms". In: *User Modeling and User-Adapted Interaction* 31.1 (2021), pp. 149–181 (cit. on pp. 5, 12, 13, 38, 39, 43, 47, 67, 74, 75).

[11] M. F. Dacrema et al. "A troubling analysis of reproducibility and progress in recommender systems research". In: *ACM Transactions on Information Systems (TOIS)* 39.2 (2021), pp. 1–49 (cit. on pp. 5, 12, 13, 47, 74).

[12] F. Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65 6 (1958), pp. 386–408 (cit. on p. 7).

[13] N. Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html (cit. on p. 7).

[14] A. Y. Ng. "Feature Selection, L₁ vs. L₂ Regularization, and Rotational Invariance". In: *Proceedings of the Twenty-First International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: Association for Computing Machinery, 2004, p. 78. ISBN: 1581138385. DOI: 10.1145/1015330.1015435. URL: https://doi.org/10.1145/1015330.1015435 (cit. on p. 7).

[15] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by F. Bach and D. Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html (cit. on p. 7).

[16] D. Bahdanau, K. Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2015) (cit. on p. 7).

[17] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (cit. on p. 7).

[18] A. Vaswani et al. "Attention Is All You Need". In: *arXiv* (2017), p. 1706.03762v5 (cit. on pp. 7, 76).

[19] J. Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv:1810.04805 [cs]* (2019) (cit. on p. 8).

[20] T. Baltrušaitis, C. Ahuja, and L.-P. Morency. *Multimodal Machine Learning: A Survey and Taxonomy*. 2017. arXiv: 1705.09406 [cs.LG] (cit. on pp. 8, 9).

[21] *Word2Vec*. URL: https://www.katacoda.com/basiafusinska/courses/nlp-with-python/embeddings/assets/word2vec.png (visited on 02/25/2020) (cit. on p. 9).

[22] *Embeddings*. URL: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/blocks/text-embedding (visited on 02/25/2020) (cit. on p. 10).

[23] T. Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *arXiv:1301.3781 [cs]* (2013) (cit. on pp. 10, 34).

[24] Y. Feng et al. *Deep Session Interest Network for Click-Through Rate Prediction*. 2019. arXiv: 1905.06482 [cs.IR] (cit. on pp. 10, 11).

[25] *SSE-PT: Sequential Recommendation Via Personalized Transformer*. New York, NY, USA: ACM, 2020, pp. 328–337 (cit. on pp. 11, 76).

[26] W.-C. Kang and J. McAuley. *Self-Attentive Sequential Recommendation*. 2018. arXiv: 1808.09781 [cs.IR] (cit. on p. 11).

[27] B. Hidasi et al. "Session-based recommendations with recurrent neural networks". In: *arXiv preprint arXiv:1511.06939* (2015) (cit. on pp. 11, 13, 39, 43).

[28] L. Wu et al. "Stochastic Shared Embeddings: Data-driven Regularization of Embedding Layers". In: *NeurIPS*. 2019 (cit. on p. 11).

[29] W.-L. Hsiao and K. Grauman. *Creating Capsule Wardrobes from Fashion Images*. 2018. arXiv: 1712.02662 [cs.CV] (cit. on p. 11).

[30] X. Dong et al. "Personalized Capsule Wardrobe Creation with Garment and User Modeling". In: *Proceedings of the 27th ACM International Conference on Multimedia*. MM '19. Nice, France: Association for Computing Machinery, 2019, pp. 302–310. ISBN: 9781450368896. DOI: 10.1145/3343031.3350905. URL: https://doi.org/10.1145/3343031.3350905 (cit. on pp. 11, 12).

[31] C. Ma et al. "Probabilistic Metric Learning with Adaptive Margin for Top-K Recommendation". In: Aug. 2020, pp. 1036–1044. DOI: 10.1145/3394486.3403147 (cit. on pp. 13, 23).

[32] C.-K. Hsieh et al. "Collaborative Metric Learning". In: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. Perth, Australia: International World Wide Web Conferences Steering Committee, 2017, pp. 193–201. ISBN: 9781450349130. DOI: 10.1145/3038912.3052639. URL: https://doi.org/10.1145/3038912.3052639 (cit. on p. 13).

[33] R. Li et al. *Towards Deep Conversational Recommendations*. 2019. arXiv: 1812.07617 [cs.LG] (cit. on pp. 16, 22).

[34] F. Radlinski et al. "Coached Conversational Preference Elicitation: A Case Study in Understanding Movie Preferences". In: *Proceedings of the Annual SIGdial Meeting on Discourse and Dialogue*. 2019 (cit. on p. 16).

[35] A. Kovashka, D. Parikh, and K. Grauman. "WhittleSearch: Interactive Image Search with Relative Attribute Feedback". In: *International Journal of Computer Vision* 115.2 (Apr. 2015), pp. 185–210. ISSN: 1573-1405. DOI: 10.1007/s11263-015-0814-0. URL: http://dx.doi.org/10.1007/s11263-015-0814-0 (cit. on p. 17).

[36] J. Habib, S. Zhang, and K. Balog. "IAI MovieBot". In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (Oct. 2020). DOI: 10.1145/3340531.3417433. URL: http://dx.doi.org/10.1145/3340531.3417433 (cit. on p. 17).

[37] R. Hadsell, S. Chopra, and Y. Lecun. "Dimensionality Reduction by Learning an Invariant Mapping". In: Feb. 2006, pp. 1735–1742. ISBN: 0-7695-2597-0. DOI: 10.1109/CVPR.2006.100 (cit. on pp. 17–19).

[38] K. Weinberger, J. Blitzer, and L. Saul. "Distance Metric Learning for Large Margin Nearest Neighbor Classification". In: vol. 10. Jan. 2006 (cit. on pp. 17–19).

[39] K. Musgrave, S. Belongie, and S.-N. Lim. "A Metric Learning Reality Check". In: *arXiv* (2020), p. 2003.08505v3 (cit. on pp. 18, 24, 42, 47, 50).

[40] D. Bonadiman, A. Kumar, and A. Mittal. "Large Scale Question Paraphrase Retrieval with Smoothed Deep Metric Learning". In: *arXiv:1905.12786 [cs]* (2019) (cit. on pp. 19, 36, 37).

[41] *Cross-Modal Subspace Learning with Scheduled Adaptive Margin Constraints*. New York, NY, USA: ACM, 2019, pp. 75–83 (cit. on pp. 19, 20, 23, 38, 76).

[42] *Adaptive Temporal Triplet-loss for Cross-modal Embedding Learning*. New York, NY, USA: ACM, 2020, pp. 1152–1161 (cit. on pp. 20, 21, 23, 38, 76).

[43] *Sampling Matters in Deep Embedding Learning*. IEEE, 2017, pp. 2859–2867 (cit. on pp. 21, 38).

[44] *#TagSpace: Semantic Embeddings from Hashtags*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, pp. 1822–1827 (cit. on pp. 30, 31).

[45] S. Wang et al. "A Survey on Session-based Recommender Systems". In: *arXiv:1902.04864 [cs]* (2021) (cit. on p. 31).

[46] RetailRocket. *RetailRocket Dataset*. 2017. URL: https://www.kaggle.com/retailrocket/ecommerce-dataset (visited on 09/24/2020) (cit. on pp. 42, 43).

[47] ACM and Y. GmbH. *ACM Recsys Challenge 2015*. 2015. URL: https://2015.recsyschallenge.com/challenge.html (visited on 02/25/2020) (cit. on p. 42).

[48] *Recsys challenge 2015 and the yoochoose dataset*. Vol. Proceedings of the 9th ACM Conference on Recommender Systems. 2015, pp. 357–358 (cit. on p. 42).

[49] *Recurrent Neural Networks with Top-k Gains for Session-based Recommendations*. New York, NY, USA: ACM, 2018, pp. 843–852 (cit. on p. 43).

[50] *Neural attentive session-based recommendation*. Vol. Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. 2017, pp. 1419–1428 (cit. on p. 43).

[51] *STAMP*. New York, NY, USA: ACM, 2018, pp. 1831–1839 (cit. on p. 43).

[52] *Performance comparison of neural and non-neural approaches to session-based recommendation*. New York, NY, USA: ACM, 2019, pp. 462–466 (cit. on pp. 43, 47, 49).

[53] *Improved recurrent neural networks for session-based recommendations*. Vol. Proceedings of the 1st workshop on deep learning for recommender systems. 2016, pp. 17–22 (cit. on p. 43).

[54] *Long-tail session-based recommendation*. Vol. Fourteenth ACM conference on recommender systems. 2020, pp. 509–514 (cit. on p. 45).

[55] *Recommender systems at the long tail*. Vol. Proceedings of the fifth ACM conference on Recommender systems. 2011, pp. 1–6 (cit. on p. 45).

[56] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008 (cit. on pp. 47–50).

[57] K. Latha. *Experiment and Evaluation in Information Retrieval Models*. CRC Press, 2017, p. 282 (cit. on p. 47).

[58] D. Garg et al. "Sequence and time aware neighborhood for session-based recommendations: Stan". In: *Proceedings of the 42nd ...* (2019) (cit. on p. 51).

[59] F. Bianchi, B. Yu, and J. Tagliabue. "BERT Goes Shopping: Comparing Distributional Models for Product Representations". In: *arXiv* (2020), p. 2012.09807v1 (cit. on p. 76).

[60] Q. Chen et al. "Behavior Sequence Transformer for E-commerce Recommendation in Alibaba". In: *arXiv* (2019), p. 1905.06874v1 (cit. on p. 76).

[61] J. Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv:1810.04805 [cs]* (2019) (cit. on p. 76).

[62] C. Liu et al. "Non-invasive Self-attention for Side Information Fusion in Sequential Recommendation". In: *arXiv preprint arXiv ...* (2021), p. 2103.03578v1 (cit. on p. 76).

[63] F. Sun et al. *BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer*. 2019. arXiv: 1904.06690 [cs.IR] (cit. on p. 76).