UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

**Evaluation of Time-Aware Recommender Systems
Techniques for Neighborhood-Based Models in
Session-Based Recommendations**

Matheus Naoto Shimura Takata

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências
de Computação e Matemática Computacional (PPG-CMC)

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Matheus Naoto Shimura Takata

Evaluation of Time-Aware Recommender Systems Techniques for Neighborhood-Based Models in Session-Based Recommendations

Master dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. Marcelo Garcia Manzato

USP – São Carlos
April 2021

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

T136e Takata, Matheus Naoto Shimura
Evaluation of Time-Aware Recommender Systems
Techniques for Neighborhood-Based Models in Session-
Based Recommendations / Matheus Naoto Shimura
Takata; orientador Marcelo Garcia Manzato. -- São
Carlos, 2021.
88 p.

Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2021.

1. Sistemas de Recomendação. 2. Sistemas de
Recomendação Baseada em Sessão. 3. Modelos de
Vizinhança. 4. Contextos Temporais. I. Manzato,
Marcelo Garcia, orient. II. Título.

Matheus Naoto Shimura Takata

**Avaliação de técnicas de sistemas de recomendação
sensíveis ao tempo em modelos de vizinhança para
recomendação em sessão**

Dissertação apresentada ao Instituto de Ciências
Matemáticas e de Computação – ICMC-USP,
como parte dos requisitos para obtenção do título
de Mestre em Ciências – Ciências de Computação e
Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e
Matemática Computacional

Orientador: Prof. Dr. Marcelo Garcia Manzato

USP – São Carlos
Abril de 2021

ACKNOWLEDGEMENTS

Gostaria de agradecer à minha família, que sempre me deu suporte durante a minha vida acadêmica. Também gostaria de agradecer aos meus amigos que me apoiaram durante esta jornada, tanto nos bons momentos como nos momentos difíceis. Além disso, gostaria de agradecer aos colegas de academia pelas trocas de conhecimento que enriqueceram a minha experiência durante o desenvolvimento deste trabalho.

*“Minoru hodo
koube wo tareru
inaho kana.”
(Provérbio Japonês)*

RESUMO

TAKATA, M. N. S. **Avaliação de técnicas de sistemas de recomendação sensíveis ao tempo em modelos de vizinhança para recomendação em sessão.** 2021. 85 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

Trabalhos tradicionais em Sistemas de Recomendação focam em trazer recomendações personalizadas de novos itens de interesse para os usuários de uma base de dados. Porém, com o crescimento de plataformas de *stream* e de serviços oferecidos online, produzir recomendações focadas na sessão atual do usuário, levando em conta o seu contexto, se tornou uma grande área de interesse para este setor de pesquisa. A área de Sistemas de Recomendação Baseada em Sessão foca em recomendar itens para uma sessão de usuário. Quando o usuário possui um histórico na plataforma, é possível utilizar sessões anteriores para inferir suas preferências. Porém, plataformas também podem atrair e fidelizar usuários recomendando itens de interesse para usuários anônimos, muitas vezes transeuntes, sendo que o sistema possui apenas os dados adquiridos da sessão atual para fazer recomendações.

Este trabalho investiga o uso de contextos temporais em Sistemas de Recomendação Baseada em Sessão, com o foco em recomendações para usuários anônimos, utilizando modelos de vizinhança, que estão entre os modelos do estado-da-arte nesta tarefa. Para isso, foi realizado um estudo aprofundado dos modelos de vizinhança existentes e como a aplicação de diversos contextos a esses algoritmos influencia o desempenho deles. Além disso, este trabalho provê também introspecções sobre as interações realizadas em uma bases de dados.

Palavras-chave: Sistemas de Recomendação, Sistemas de Recomendação Baseada em Sessão, Modelos de Vizinhança, Contextos Temporais.

ABSTRACT

TAKATA, M. N. S. **Evaluation of Time-Aware Recommender Systems Techniques for Neighborhood-Based Models in Session-Based Recommendations.** 2021. 85 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2021.

Traditional works in Recommender Systems focus on making personalized recommendations of new items of interest to users of a database. However, with the growth of streaming platforms and services offered online, generating recommendations for the current user session, considering his/her context, became a big area of interest in the area. Session-Based Recommender Systems focus on recommending items to a user session. When the user has a history of interactions with the platform, previous sessions can be used to infer the user's preference. Nevertheless, these platforms can also attract and regularize users by recommending items of interest to anonymous users, usually passerbies, who have only the information available of the current session to make recommendations. This work investigated the use of temporal contexts in Session-Based Recommender Systems, focusing in anonymous sessions recommendations, using neighborhood-based models, which are among the state-of-the-art models in this task. For this to happen, we performed a deep analysis of the existing neighborhood-based models and analyzed how the application of various contexts in these algorithms influences their performance, as well as providing insights about the interactions made in a given dataset.

Keywords: Recommender Systems, Session-based Recommender Systems, Neighborhood-Based Models, Temporal Contexts.

LIST OF FIGURES

Figure 1 – Hierarchical structure of session data and specific challenges in SBRS.Extracted from (WANG; CAO; WANG, 2019)	40
Figure 2 – The architecture of the S-KNN model, from the training stage to the recommendation list to an anonymous user.	44
Figure 3 – Illustration of the first proposed extension for the S-KNN model. The content inside the blue box represents the additional data structure created to map sessions by context, which is used by the sampling step to filter out sessions that did not happened in the same context. The contexts used are described in section 4.3.	55
Figure 4 – Illustration of the second proposed extension for the S-KNN model. The extension proposed in the architecture is highlighted by the blue box, where the proposed scoring methods uses the indexing of sessions per context to lower computational costs. The contexts used are described in section 4.3.	56
Figure 5 – Number of interactions by hour for the datasets RetailRocket, RSC15 and Nowplaying, respectively.	63
Figure 6 – Number of access by day of the week for datasets Retailrocket, RSC15 and Nowplaying.	63
Figure 7 – Number of average sampled sessions per recommendations vs maximum sample size. On the left, the chart for the Retairocket dataset. On the right, the chart values for the dataset NowPlaying.	65

LIST OF TABLES

Table 1	– Example of a matrix used by a traditional collaborative filtering method to represent the interactions between users and items.	26
Table 2	– Example of a dataset prepared used for SBRS studies.	41
Table 3	– Representations of sessions s_n in SBRS. Sessions are represented by lists of items i_m that users interacted within a given session. In the illustration, the task of the recommender is to predict the next item that u_5 would interacted, characterized by “?”.	42
Table 4	– On the left, representations of sessions s_n in SBRS. Sessions are represented by lists of items i_m that users interacted within a given session. In the illustration, the task of the recommender is to predict the next item that u_5 would interacted, characterized by “?”. Also, the table on the right represents a structure that can easily retrieve the timestamp of the last interactions of the sessions. . . .	45
Table 5	– Contexts extracted from the example log.	55
Table 6	– Statistics of datasets RetailRocket, RSC15 and Nowplaying after filtering sessions with at least 2 interactions and items with at least 5 interactions. . .	63
Table 7	– Average of sample sessions for each dataset, with $k=50$ and $m = 100$ for Retailrocket and Nowplaying, and $k = 500$ and $m = 1000$ for RSC15.	66
Table 8	– Results of different contexts mentioned in section 4.3 applied to S-KNN as a sampling strategy for the dataset RetailRocket. In this dataset, dividing the dataset in two contexts yielded the best results, with the division by weekdays and weekends having the best performance, followed by dividing the interactions by morning/evening and even/odd hours. The divisions by period of the day, weekdays, and hours had worse performances, as they gradually sliced the dataset into more parts.	67
Table 9	– Results of different contexts mentioned in section 4.3 applied to S-KNN as a sampling strategy for the dataset Nowplaying. In this dataset, the performance of each proposed context followed the same pattern as Table 8, with the exception that in this table, the context even/odd hours performed slightly better than the morning/night context.	67
Table 10	– Results of different contexts mentioned in section 4.3 applied to S-KNN as a sampling strategy for the dataset Nowplaying. In this dataset, the performance of each proposed context followed the same pattern as Table 9.	67

Table 11 – Results of different contexts mentioned in section 4.3 applied to S-KNN as a sampling strategy for the dataset NowPlaying.	68
Table 12 – Results of different contexts mentioned in section 4.3 applied to S-KNN as a sampling strategy for the dataset Retailrocket.	69
Table 13 – Average results of different contexts mentioned in section 4.3 applied to S-KNN as a sampling strategy for the dataset RSC 15.	70
Table 14 – Results of different contexts applied to S-KNN as a scoring method for the dataset Nowplaying.	72
Table 15 – Results of different contexts applied to S-KNN as a scoring method for the dataset Retailrocket.	72
Table 16 – Average results of different contexts applied to S-KNN as a scoring method for the dataset RSC15.	73
Table 17 – Results of different contexts comparing pre-filtering and post-filtering strategies for RetailRocket.	74
Table 18 – Results of different contexts comparing pre-filtering and post-filtering strategies for NowPlaying.	75
Table 19 – Average results of different contexts comparing pre-filtering and post-filtering strategies for RSC15.	75

LIST OF ABBREVIATIONS AND ACRONYMS

AR	Association Rules
CARS	Context-Aware Recommender Systems
CF-SKNN	Context-Filter S-KNN
EPCSR	Equal Probability Candidate Selection at Random
HR	HitRate
I-KNN	Item-based KNN
ICF-SKNN	Inverse-Context-Frequency S-KNN
LBSN	Location-Based Social Networks
MC	Markov Chains
MRR	Mean Reciprocal Ranking
POI	Point-of-Interest
RL	list of relevant sessions
RS	Recommender Systems
S-KNN	Session-KNN
S-SKNN	Sequential Session-based KNN
SARS	Sequence-Aware Recommender Systems
SBRS	Session-Based Recommender Systems
SF-SKNN	Sequential Filter Session-based KNN
SR	Sequential Rules
STAN	Sequence and Time-Aware Neighborhood
SVD	Singular Value Decomposition
TARS	Time-Aware Recommender Systems
TARS	Time-aware Recommender System
V-SKNN	Vector Multiplication Session-Based KNN

CONTENTS

1	INTRODUCTION	21
1.1	Contextualization and Motivation	21
1.2	Objective and Hypotheses	23
1.3	Dissertation Organization	23
2	RECOMMENDER SYSTEMS	25
2.1	Traditional Recommender Systems	25
2.1.1	<i>Collaborative Filtering</i>	26
2.1.1.1	<i>Neighborhood-based methods</i>	26
2.1.1.2	<i>Matrix-Factorization methods</i>	27
2.1.1.3	<i>The Cold-Start Problem</i>	29
2.1.2	<i>Content-Based Filtering methods</i>	29
2.1.3	<i>Knowledge-Based Recommender Systems</i>	30
2.1.4	<i>Ensemble-Based and Hybrid approaches</i>	31
2.2	Context-Aware Recommender Systems	31
2.2.1	<i>Time-Aware Recommender Systems</i>	33
2.2.2	<i>Sequence-Aware Recommender Systems</i>	34
2.3	Evaluation of Recommender Systems	35
2.3.1	<i>Ranking prediction metrics</i>	35
2.3.2	<i>Beyond-accuracy metrics</i>	36
2.3.3	<i>Validation</i>	37
2.4	Final Considerations	37
3	RELATED WORK	39
3.1	Session-Based Recommender Systems	39
3.1.1	<i>Association Rules</i>	42
3.1.2	<i>Markov Chains</i>	42
3.1.3	<i>Sequential Rules</i>	42
3.1.4	<i>Neural Network models</i>	43
3.1.5	<i>Factorization models</i>	43
3.1.6	<i>Neighborhood-based Models</i>	43
3.1.6.1	<i>S-KNN extensions</i>	47
3.2	Applications of TARS approaches in Session-Based Recommendations	50

3.2.1	<i>TARS approaches with Neighborhood-based models</i>	50
3.3	Final Considerations	51
4	PROPOSAL	53
4.1	Project Justification	53
4.2	Proposed Models	54
4.2.1	<i>Context as a pre-filtering strategy</i>	54
4.2.2	<i>Context as a post-filtering strategy</i>	56
4.2.3	<i>Scoring methods</i>	56
4.3	Context Extraction	58
4.4	Final Considerations	58
5	EVALUATION	61
5.1	Methodology	61
5.1.1	<i>Datasets</i>	61
5.1.2	<i>Evaluation</i>	64
5.1.3	<i>Metrics</i>	64
5.1.4	<i>Auxiliary Resources</i>	64
5.2	Experiments	64
5.2.1	<i>Using context to sample sessions</i>	64
5.2.1.1	<i>Efficacy of temporal contexts as pre-filtering strategies</i>	66
5.2.1.2	<i>Discussion</i>	68
5.2.2	<i>Context in score functions</i>	71
5.3	Pre-filter vs Post-Filtering approaches	71
5.4	Final Considerations	73
6	CONCLUSIONS AND FUTURE WORKS	77
6.1	Summary	77
6.2	Contributions	78
6.3	Limitations	79
6.4	Future Works	79
	BIBLIOGRAPHY	81

INTRODUCTION

1.1 Contextualization and Motivation

The Web 2.0 was the period in which users became not only consumers of information but also producers of information, providing content such as blog posts, videos, reviews for products and services on the myriad of platforms available on the Web. As the relevance of the Internet increased in people's lives throughout the years, the amount of content, information, products, services, and opinions available to consume became overwhelming. In order to help users find what interests them, platforms have developed tools such as Recommender Systems (RS), that can use explicit preferences and browsing history to infer users' preferences and suggests items to be viewed or consumed.

One example of successful implementations of RS is Amazon's "people who buy this item would also buy these" recommendations, which was estimated to reach 30 percent of the page views (SMITH; LINDEN, 2017). Netflix is another platform that not only uses RS but also motivated research in this area with the Netflix Prize¹, a contest that awarded one million dollars to the team who developed an RS that could predict ratings of users 10 percent better than Netflix's RS at the time. Nowadays, RS are commonly used in a variety of domains. In Brazil we can see e-commerces such as Centauro² and Magazine Luiza³ offering recommendations as "similar products" or "buy these with these" sections in their platforms. We can also see the use of RS in streaming services such as Spotify, that has used RS to improve the model of users' preferences and recommend future songs that a user is likely to listen (BOOM *et al.*, 2018).

Most works in RS use explicit feedback from users, such as positive and negative reviews or ratings, in order to infer users' preferences. With information about what a user likes and dislikes, RS can build a profile of the users which will be used to predict the likeability of the

¹ <https://www.netflixprize.com>.

² <https://www.centauro.com.br>.

³ <https://www.magazineluiza.com.br>

users towards the items available in the database. However, this approach is not suitable when recommending items for new or anonymous users, which is a scenario that often happens in e-commerces. Also, for music and video platforms that provide streaming services, such as Spotify and Youtube, understanding the users' context through the items they are consuming in an ongoing session may be more interesting than having all the mostly likeable items grouped together, and the rise in popularity of these services also raised interest in Session-Based Recommender Systems (SBRS), which is indicated by Spotify organizing the 2018 ACM Recsys Challenge⁴, consisted in predicting the items of users playlists.

When recommending items for an ongoing session, adaptations of deep learning techniques for sequence modeling problems such as Recurrent Neural Network models have been successfully explored in (HIDASI *et al.*, 2015). Also, (Barkan; Koenigstein, 2016) has shown that relevant item representations can be generated with adaptations of Word2Vec models that uses the sequence of items consumed in sessions as sentences to be analyzed by these models. However, a recent survey (LUDEWIG; JANNACH, 2018) showed that neighborhood models can outperform these deep-learning techniques in the next-item prediction task when considering anonymous sessions, besides having a simpler and easier-to-explain recommendation process.

Since the process of finding the nearest neighbors grows exponentially as the number of sessions and items of a platform grows, systems that uses neighborhood-based models have to consider the computational costs of the recommendation process in session-based scenarios, as adapting the lists of recommendations after every interaction is a desirable feature for this scenario. (LUDEWIG; JANNACH, 2018; GUO *et al.*, 2019) have proposed models that adapted the traditional user-based KNN, the Session-KNN (S-KNN) models, by adopting sampling strategies of sessions to reduce the search space before finding the k nearest neighbors, and have showed that sampling based on finding the most recent sessions were a successful strategy. They also argued that anonymous sessions can be influenced by seasonal trends, which opens rooms for other temporal contexts to be investigated and how they influence the quality of recommendations.

One common task of SBRS is to recommend the immediately next item to an ongoing session. The minimum data required for this task are the accessed items of the ongoing sessions, and a timestamp of each interaction in order to get the order the items were accessed, or at least the order of the events. Some publicly available datasets mask the original timestamp of the interactions for privacy concerns, however, when available, temporal contexts, such as day of the week and hour of the day, can be easily extracted to be used in Context-Aware Recommender Systems techniques (ADOMAVICIUS; TUZHILIN, 2011). These techniques assumes that context plays an important role for items of interest. For example, the genre of songs people want to listen to in the morning may be different than the kind of music they want to listen to after work hours. A pre-filtering approach, in this example, could recommend

⁴ <https://recsys.acm.org/recsys18/challenge>. Accessed in January 12th, 2020.

songs for specific periods of the day, and in neighborhood-based models, it can also be used to sample sessions within the same context, alleviating the computational costs for finding the most similar neighbors, assuming that the given context influences the interactions. As a post-filtering approach, these contexts can raise the importance of the sessions that occurred in a given session. Moreover, exploring extracted contexts from timestamps allows algorithms to explore more features without platforms needing to expose more data about their items and users for SBRS tasks.

1.2 Objective and Hypotheses

The main objective of this work is to investigate the influence of different temporal contexts generated from the timestamps of the interactions in neighborhood-based models, focusing on the Session-KNN and the extensions proposed by (LUDEWIG; JANNACH, 2018). For this, we proposed two extensions of the S-KNN model with time-related contexts: a pre-filtering technique as another sampling strategy, and a post-filtering technique that can leverage the score of items of sessions that occurred in the same context.

This project was developed based on the following hypotheses:

1. Session-Based Recommender Systems can provide better recommendations with the use of additional contextual features generated from the timestamps of the ongoing session when using neighborhood-based models.
2. Temporal contexts generated from timestamps can be effectively used as a pre-filtering technique to reduce the search space used find the nearest neighbors in S-KNN models and extensions.
3. Temporal contexts generated from timestamps can improve the quality of a recommendation list by giving more relevance to sessions that occurred in the same context.

To validate these hypotheses, we conducted two sets of experiments where we evaluated and analyzed the techniques mentioned above against other extensions of the S-KNN model found in the literature.

1.3 Dissertation Organization

This Chapter presented the contextualization, motivations, objectives and hypotheses that guided the development of this work. Chapter 2 presents a literature review with fundamental concepts of this work, describing traditional approaches in the area, as well as standard evaluation techniques for this area. Chapter 3 presents an overview of Session-Based Recommender Systems, presenting focusing on works using neighborhood-based models used for SBRS, as well as

recent techniques using time-aware recommender system approaches. Chapter 4 describes the architecture used to evaluate the approaches proposed in this work. Chapter 5 reports the results of the experiments developed for this project, as well as the description of the methodology and analysis of the results. Finally, Chapter 6 summarizes this work, also describing the limitations, contributions and future works.

RECOMMENDER SYSTEMS

This chapter covers fundamental concepts of Recommender Systems that serve as the basis of the work, pointing out differences between traditional approaches and Session-Based Recommender Systems.

2.1 Traditional Recommender Systems

Traditional works in RS focuses on estimating how accurate the systems were able to guess users' tastes. In order to do that, RS approaches uses explicit feedback, such as positive/negative feedback, reviews, and ratings (e.g. a scale of one to five stars). RS rely on the history of interactions between users and items in order to make personalized recommendations. As more diverse data are made available, such as timestamp of interactions and browsing logs, RS approaches can also extend traditional models in order to enhance baseline approaches.

[AGGARWAL](#) has classified the tasks of traditional RS into two groups:

1. Predict accurate ratings for a user-item combination. This task is better suited when the RS has access to explicit feedback, which can be used to estimate the level of interest of an user towards all the items of the database. With this, a system can offer recommendations high levels of interest for their users, which helps users engage more with the platforms, enhancing their levels of satisfaction.
2. Generate a ranked list of top-k most relevant items to a user. This task has gained more relevance as works in the area have shifted from explicit feedback to implicit feedback, as the later type of feedback is more easily obtained and often more abundant.

In Section [1.1](#) we briefly introduced CF and CBF approaches. The subsections below describe the general intuition behind each approach, pointing out strengths and weaknesses of each class of algorithms.

2.1.1 Collaborative Filtering

Collaborative Filtering algorithms make recommendations assuming that like-minded users consume the same items. Traditional algorithms build users' profiles from their history of interactions, and most approaches transform these interactions into a matrix of $Users \times Items$. An example is illustrated in table 1, where users are represented by u_n and items are represented as i_m , and the ratings numerical values between 1 and 5.

Table 1 – Example of a matrix used by a traditional collaborative filtering method to represent the interactions between users and items.

	i_1	i_2	i_3	i_4	i_5	i_6	i_7
u_1	5	?	2	?	4	4	?
u_2	?	5	?	?	5	4	2
u_3	?	4	4	2	3	5	?
u_4	5	?	?	?	2	1	5
u_5	?	?	2	5	2	3	?

These matrices tend to be very sparse, since, in most cases, users will interact with only a fraction of the existing items. For this reason, one of the most common difficulties dealing with the cold-start problem, which is the inability of the recommender to suggest items due to the lack of input. This section describe common approaches used in: *neighborhood-based*, and *model-based* methods.

2.1.1.1 Neighborhood-based methods

Neighborhood-based methods were among the earliest algorithms developed for collaborative filtering (AGGARWAL, 2016e). These methods are divided in:

- user-based approaches, where the algorithm uses the interactions of similar users on a particular item in order to predict how a given user would rate the item;
- item-based approach, in which the algorithm uses the items with most similar interactions to estimate the interest of the user for a given item.

When the number of users is bigger than the number of items, which is commonly the case in a successful system, item-based approaches usually perform better than user-based counterparts. Moreover, item-based methods are more stable because two items are more likely to have a larger number of users who have co-rated them (AGGARWAL, 2016e).

For a prediction task, a simple user-based neighborhood algorithm, the rating $\hat{r}_{ui}$ of an item would be given as: considering a user u , $N(u)$ the neighbors of u , users $v \in N(u)$ and w the

calculated similarity between u and v :

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i(u)} w_{uv} \times r_{vi}}{\sum_{v \in N_i(u)} |w_{uv}|} \quad (2.1)$$

Similarity between users can be calculated using Pearson correlation, cosine distance using the item dimension as vectors, among other approaches. In the item-based approach, the model uses as a representation of items all ratings users have given to them. The similarity between items are usually calculated by the cosine similarity, and a predicted rating of an item i for a user u will be given by the weighted average of the top- k similar items.

Neighborhood-based methods are popular because of their simplicity and explainability. Although the cost of updating the model is low, the computational costs of making a recommendation are high, since the calculations are done in the prediction stage. Another problem of these approaches is the space cost, which grows fast as the number of users and items rises. To soften these problems, some works have applied clustering techniques (COSTA; MANZATO; CAMPELLO, 2016) to reduce the search space before finding the nearest neighbors; other approaches used dimensionality-reduction techniques such as creating item embeddings (Barkan; Koenigstein, 2016) from the interactions. LUDEWIG; JANNACH used auxiliary data structures and applied strategies to sample user sessions in order to reduce the computational costs of the prediction phase, and this method will be discussed later in Chapter 3.

Another problem Neighborhood-based methods face is data sparsity. Because there are hardly large groups of users that interact with the same items, it can be challenging to find k -neighbors with high similarity (AGGARWAL, 2016e; NING; DESROSIERS; KARYPIS, 2015). So, in order to avoid data sparsity, Some approaches have used graph-based methods (PALUMBO; RIZZO; TRONCY, 2017) where the matrix of interactions is treated as a graph, and representation of items and users are extracted with Computer Network techniques. GUO *et al.* also treated the interactions as a bi-partited graph to calculate the similarity between sessions using the degree of the nodes. This method will also be better discussed in chapter 3.

Since SBRS tasks usually do not involve explicit numerical ratings, neighborhood-based models must be adapted to work in these scenarios. The user-based approach was adapted and named Session-Based KNN (JANNACH; LUDEWIG, 2017), whose extensions were made based on the sequence of the interactions (LUDEWIG; JANNACH, 2018), and the item-based KNN will recommend the next item by finding the most similar items to the last accessed item by the session. These adaptations are discussed in Section 3.1.6.

2.1.1.2 Matrix-Factorization methods

Matrix-Factorization (MF) models have been the state-of-the-art models since the Netflix Prize. The models proposed by (KOREN; BELL; VOLINSKY, 2009), also called Singular Value Decomposition (SVD) models, create a latent space that tries to explain ratings by characterizing

both items and users on factors automatically inferred from user feedback (KOREN; BELL, 2015). Unlike neighborhood-based approaches, the training stage is clearly separated from the prediction stage. In the training stage, MF methods create a latent space that tries to find the characteristics that attract users, as well as characterizing items with the same latent factors.

In a predictive task, CF models soften cold-start problems by using a baseline predictor that captures most of the signals of the data and serves as a good initial estimator in cold-start scenarios (KOREN; BELL, 2015). This baseline can be initiated as follows: denoting u for user, i for item, μ for the overall average rating, b for bias, a simple estimator function would predict a rating r_{ui} that a user u would give to an item i , where the user bias refers to how the ratings of a user u differs from the mean, and item bias is how the ratings of a given item i differs from the mean. The equation becomes (KOREN; BELL, 2015):

$$r_{ui} = \mu + b_u + b_i \quad (2.2)$$

MF models then proceed the factorization of the matrix of interactions, generating a vector $q_i \in \mathbb{R}$ that is associated to each item i , while also generating a vector $p_u \in \mathbb{R}$ associated to each user u . After vectors p and q are generated, the prediction of a rating would be generated with:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \quad (2.3)$$

Some examples of MF algorithms are the SVD model (KOREN; BELL; VOLINSKY, 2009), where the algorithms initialize the latent factor vectors p_u and q_i with random values and adjust these vectors' values with the nonzero values from the matrix of interactions, adjusting errors with Stochastic Gradient Descent; and Non-negative MF, that works better when implicit feedback is available, because in these scenarios recorded interactions are often considered as positive feedback, but negative feedback cannot be easily inferred.

Incorporating other sources of data into MF models can also lead to improvements into the representations of items and users. The SVD model has inspired variations such as SVD++, that incorporates implicit feedback into the representations. gSVD++ (MANZATO, 2013) extends the SVD++ model by incorporating metadata information to improve cold-start scenarios. Time-SVD++ (KOREN, 2010a) were one of the earliest works using the sequence of interactions to improve recommendations, extending the SVD++ model by using time as a variable to capture driftings in user preferences. However, traditional MF models do not leverage the context of the ongoing session when recommending items. Section 3.1.5 describes adaptations of these models for session-based recommendations.

2.1.1.3 The Cold-Start Problem

Recommending items to anonymous sessions using a CF approach can be seen as a special case of the new user problem, since no history of interactions is present for the ongoing session, nor can it be linked the current user to be used into following sessions. In general, CF approaches tend to improve their accuracy as the number of interactions rises. In this way, the model is able to both understand users' preferences and how the items are perceived from the crowd. In the session-based scenario, more interactions will benefit the models in a similar way: for a neighborhood-based model, for example, more sessions will increase the likelihood of the model to find similar sessions to be used for recommendations; and more items accessed within a session can help the algorithm to understand the context of interaction of the current session.

2.1.2 Content-Based Filtering methods

Content-Based Filtering (CBF) algorithms are approaches that tries to match items with characteristics that are similar to the profile the system builds for the users. When plenty of description about items is available, such as categories, tags, prices, systems can leverage this information to build good item representations. When reviews, comments and queries from users are available, RS can also use this data to construct accurate representations of items and understand the preferences of the users, which will be used to suggest items that better suits their preferences.

The high level architecture of a CBF approach can be divided into three components (GEMMIS *et al.*, 2015):

1. **Content Analyzer:** This module generates representations of items from the information made available to the system. Top-down approaches integrates external knowledge, such as taxonomies, ontologies and machine-readable dictionary in order to capture the semantics of the information used to represent items and build user profiles. Bottom-up models, on the other hand, performs better when large corpora is available to analyze complex relations between words, generating dense vector representations of words that will be used to build content representations.
2. **User Profile Learner:** This component will analyze stored feedback from users to infer the features that represent a user's interests. When there isn't a positive or negative feedback available, the learning can also use Sentiment Analysis techniques to infer the user's position about the item (D'ADDIO; DOMINGUES; MANZATO, 2017).
3. **Filtering Component:** In this component, the algorithm exploits the user profile to suggest relevant items by matching it against the collection of items.

(GEMMIS *et al.*, 2015) points out that novel research works in CBF have shifted from a keyword-based to a concept-based representation of items and user profiles, taking advantage of advances in Natural Language Processing and Semantic Technologies. They separate these approaches in top-down and bottom-up semantic approaches. In top-down semantic approaches, information from other sources are incorporated in order to build item representations, such as Linked Open Data (NOIA *et al.*, 2012), encyclopedic knowledge sources, such as Wikipedia (MUSTO *et al.*, 2016), or incorporate ontological knowledge. On the other hand, bottom-up semantic approaches infer the semantics of the content the item has based on distributional models, such as Word2vec (MIKOLOV *et al.*, 2013b), where the meaning of the content is inferred based on the occurrences it appears in the database.

Since CF approaches does not need to rely on users' feedback in order to make recommendations, these algorithms do not suffer from cold-start problems. However, the quality of the recommendations made by these approaches are limited by the data available to characterize items. For example, if the items are characterized with only a few features, the algorithms will not be able to analyze and differentiate items, which can compromise the ability of the recommender to choose the most interesting items to the users. Another distinction between CF and CBF approaches is that CF models can explore the variety of users' preferences to find more intrinsic relationships among items that goes beyond metadata descriptions (AGGARWAL, 2016a), while CBF algorithms may suffer from overspecialization of recommendations, a problem that happens when the recommender is not able to recommend items with a more diverse set of characteristics.

Developing human-understandable representations of items and users allows algorithms to be more explainable and researches to develop more in-depth analysis about the features available. Although the intuition behind CBF approaches is more suitable to anonymous session-based recommendations, since this problem is essentially a cold-start scenario. Datasets publicly available for SBRS tasks often encode their metadata in a non-human readable way due to privacy concerns, limiting the potential of the use of CBF approaches in this area.

2.1.3 Knowledge-Based Recommender Systems

For domains where the cost of consumption is high, such as car and house shopping, explicit feedback is more scarce, and recommendations need to be more tailored for the users. In these scenarios, Knowledge-based RS are more indicated, as these approaches use explicitly asks for the users' preferences and restrictions towards items, which are used to match items in the database that match these preferences. Main methods can be divided in *constrained-based* and *case-based* methods, where the first method uses user specifications in order to find relevant items, and the latter acts on finding similar items from the item specified from the user (AGGARWAL, 2016d).

SBRS are often used as non-interrupting approaches to user navigation through the consumption of the items. Although they could be used to improve the start of browsing, these

approaches have not gotten the attention of the area.

2.1.4 Ensemble-Based and Hybrid approaches

All the approaches mentioned so far have their own sets of strengths and weaknesses, which make them limited in isolation. However, as multiple data sources become available to the systems, different strategies can be combined in order to generate better recommendations. (AGGARWAL, 2016b) have classified these strategies in:

- Ensemble design: this approach combine the results of different off-the-shelf algorithms into a more robust recommendation list. An example of this approach is a model developed by (ZHANG; CHENG, 2016), that used Word2vec algorithms (MIKOLOV *et al.*, 2013a) to create item embeddings from unstructured data and use a SVD model to re-rank the list of recommendations generated.
- Monolithic design: Various data types are used in order to create an integrated recommendation algorithm. For example, (D’ADDIO; DOMINGUES; MANZATO, 2017) used a CBF approach to generate user and item profiles from reviews using feature extraction and sentiment analysis techniques, and leveraged simplicity, efficiency, stability of a *item-knn* CF algorithm to perform the recommendations.
- Mixed design: Multiple algorithms are combined in a black-box approach. For example, the gSVD++ algorithm (MANZATO, 2013) enhances a Matrix Factorization model by incorporating items’ metadata when available, which was proven to improve accuracy measures.

2.2 Context-Aware Recommender Systems

Traditional RS approaches assume that users’ interests are stable when recommending items. However, interests can be highly influenced by the context of the user in the moment of the interaction. For example, people can like a certain genre of music for parties and another genre to listen to before they go to sleep, or they can have a certain opinion about a movie when they watched it in their teenage years, and change it as they get older. RS techniques that incorporate the context of the users during the recommendation process are called Context-Aware Recommender Systems (CARS).

Context is defined by (DEY, 2001) as:

any information that can be used to characterize the situation of an entity. Entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

([DOURISH, 2004](#)) categorized contexts into *representational* and *interactional* views. In the *representational* view, context is understood as an outlined, stable form of information that is independent from the underlying activity. So, when building a CARS using a representational view, system designers have to decide *a priori* which contexts will be captured by the system and how they will be used in the recommendation process. For example, for Location-Based Social Networks (LBSN), the geolocation of the user when interacting with the platform is a key context to be captured by the system, so that an RS can recommend places that are feasible for the user to go. In a Time-aware Recommender System (TARS), the time of the interaction can be used to make recommendations based on what other users have done in the same context, which can be classified as the period of the day, or by the season of the year, for example.

On the other hand, in the *Interactional* view, contextuality is a dynamic property that arises from the activity ([DOURISH, 2004](#)). This way of interpreting context explain one of the main challenges for recommender systems, which is that the system should not rely only on the profile that they build by understanding their preferences in the past, but they should also consider the current context of the user in the moment of the interaction, knowing that their preferences in the current session can also change as the interactions happen. For example, a user that consistently listened to rock songs may prefer listening to acoustic songs in a given session because they played an acoustic version of a song recorded by their favorite artist. In the RS area, sequence-aware recommender systems uses this interpretation of context by trying to understand the state of the mind of the user by the order of the interactions.

When encoding the contextual information into the models, [ADOMAVICIUS; TUZHILIN](#) simplified that the rating function R used by traditional approaches changed from

$$R : User \times Item \rightarrow Rating \quad (2.4)$$

to

$$R = User \times Item \times Context \rightarrow Rating \quad (2.5)$$

In other words, CARS adds another dimension of contextual information into the rating function used by traditional models, transforming the $User \times Item$ matrix into a tensor, and the strategies used to encode the context dimension can be separated into three different categories: contextual pre-filtering, contextual post-filtering and contextual modeling.

In the contextual pre-filtering, systems can use a representational view to separate the dataset into contexts, in order to explore a smaller yet meaningful amount of examples in order to yield recommendations. For example, [BALTRUNAS; AMATRIAIN](#) used context of interactions to build *micro-profiles* of the users for different contexts, so that the algorithm could give different recommendations according to the time of the interaction. [LUDEWIG; JANNACH](#) used the most

recent sessions to sample the dataset in order to reduce the search space of a neighborhood-based model, which led to accuracy gains.

In the contextual post-filtering approach, context can be used as a filter to remove unwanted items for the given context. For example, the system can avoid suggesting items for Halloween after the period passed. Another way to explore this strategy is by enhancing the value of the items according to the context, for example, raise the value of nearby places, in the case of a LBSN (LI *et al.*, 2017b).

Finally, the contextual modeling approach encodes context into the recommendation function as an explicit predictor of a user's rating. KOREN gave higher weight for more recent feedback, incorporating into a Matrix-factorization model, successfully capturing drifting of interests of the users. Contextual variables were also explored to modify similarity functions for neighborhood-based models, which will be described in Section 3.1.6.

2.2.1 Time-Aware Recommender Systems

Time-Aware Recommender Systems (TARS) is a widely used approach for CARS. Since timestamps of the interactions are non-invasive information that can be collected from the users, RS can use time as context to improve the recommendation process by encoding time in a representational view, separating the interactions by the period of the day, for example, or in an interactive way, by identifying the order of interactions. AGGARWAL have separated TARS approaches into the following categories:

- *recency-based* approaches, which tend to give higher weight to the feedback received in the latest interactions. These approaches are divided into *decay-based* and *window-based* models. In the decay-based approach, time is often used in a weighting function to assign lower weights for older interactions. For example, GARG *et al.* applied the weighting function into the similarity calculations among sessions, attributing lower values the further apart the sessions occurred. In windows-based approaches, a window of time is determined so that only sessions that occurred within that window will be used in the recommendation process. LUDEWIG; JANNACH have used a window-based approach to reduce the search space of a neighborhood model by sampling the most recent sessions before finding the nearest neighbors.
- *periodic-based* approaches analyze the interactions based on representational contexts, that will be used as either pre-filtering or post-filtering strategies. Besides (BALTRUNAS; AM-ATRIAIN, 2009), DOMINGOS *et al.* also explored the influence of these representational views into a neighborhood-based model used in session-based recommendations.
- *models that explicitly use time as an independent variable* are models that encode items and times into the same representation. KOREN incorporated time into a factorization

model so that the representations of items can capture the stability of the popularity over shorter periods of time. Moreover, the model considers the variance of the ratings of the users through time.

AGGARWAL also used the term “Discrete Temporal Models” to reference models that use time to order or group the logs of interactions (ZHOU *et al.*, 2007). These approaches gained attention of the community because these algorithms often analyze the recommendation problem in a more realistic way, since the intuition behind these models is that short-term interests of the users can change between interactions within a session, so RS have also take that into account, even though their long-term preferences are understood. Next section will briefly discuss these approaches.

2.2.2 Sequence-Aware Recommender Systems

Sequence-Aware Recommender Systems (SARS) are systems that take the sequence of actions of a user in a platform in order to recommend the next interactions. Most SARS extend the traditional CF approach by incorporating into the models time-related information is used to analyze how long and short-term preferences affects the users in a given session.

Since the order of the interactions is a key factor in these approaches, the main tasks these algorithms aim to solve also differs from the traditional approaches. SARS approaches assumes that users’ interests rapidly evolves across sessions, whereas traditional approaches tends to yield more stable recommendations according to their general preferences. QUADRANA; CREMONESI; JANNACH divided the tasks of SARS models into:

- Predict the immediate next item of interest: in this task, the goal of the recommender is to suggest the next item to be consumed, given all items already consumed within a session. Common domains are: suggesting the next point-of interest to a user, suggest the next song to be listened, the next news to be read, or the next product to be searched or bought.
- Predict the next n items: this task is more used in music-streaming sessions, such as an automatic radio station. By presenting the next group of songs to be played, users can have a better notion if the system understands what they want at the moment, and adapt it accordingly.
- Predict the next session: this task can also be seen in music-streaming services such as Spotify¹, where they suggest playlists of the week, based on latest activities.

Although SBRS can be seen as a subset of SARS, most works in this area rely on multiple sessions of the user to recommend items to an ongoing session. This work, however, focuses on

¹ [spotify.com](https://open.spotify.com)

recommending items for anonymous sessions, which cannot rely on inter-session information. Since most techniques used in SARS can be applied in SBRS with adaptations, we will discuss the main techniques used both in SARS and SBRS in the next section.

2.3 Evaluation of Recommender Systems

The quality of the recommendations given by a RS have been traditionally evaluated in offline settings, predicting how well the algorithms would predict past interactions of the dataset. This method of evaluation may lack the trustworthiness of the results when compared to online experiments with real users (GUNAWARDANA; SHANI, 2015), but they are easier to conduct and can quickly yield preliminary results about the prediction power of new approaches. In this section we present the traditional metrics used to evaluate models in offline settings, focusing on metrics standard metrics used to evaluate SBRS algorithms. We also describe validation techniques used to minimize concerns with bias the dataset may have.

2.3.1 Ranking prediction metrics

The evaluation of ranked lists of recommendations can be made by evaluating how high-ranked an item the user has previously interacted is in the list provided. Having a good *top-N* list of recommendations is desirable since this indicates that the system understand what the users are interested at the moment and facilitates the search process. The traditional metrics used to measure the quality of the lists of recommendations are:

- *Normalized Discounted Cumulative Gain (NDCG)*: In this metric, the relevance gain is measured by discounting the relevance of the items in a logarithmic scale according to position of the items. In other words, this is a metric to check how well does the algorithm perform against the ideal list of recommendation. This metric is better used when numeric ratings or a ranked list of preferences is available to the system. The formula for the Discounted Cumulative Gain (DCG) for a list of J items is defined as

$$DCG = \frac{1}{N} \sum_{u=1}^N \sum_{j=1}^J \frac{g_{u,i_j}}{\log_b(j+1)} \quad (2.6)$$

where i_j is the item at position j in the list. Then, NDCG can be calculated by dividing the DCG by the ideal DCG, denoted by DCG*:

$$NDCG = \frac{DCG}{DCG^*} \quad (2.7)$$

- *Precision at N (P@N)*: This metric evaluates how many of the Top-N item recommendations match with users' history on average. An average of P@N of the lists will be made

to evaluate the efficiency of the results. Being *ret* the number of matching items and *N* the number of recommended items, $P@N$ is calculated as:

$$P@N = \frac{ret}{N} \quad (2.8)$$

Both NDCG and $P@N$ metrics have been used to evaluate SBRS approaches. NDCG and a variation of $P@N$ called R-Precision were used in the RecSys Challenge 2018² to evaluate algorithms in the task of predicting the missing songs in a playlist. This project, however, focused in the task of predicting the immediate next item, so these metrics were not best suited for the task. For the task mentioned, the established metrics used are (LUDEWIG; JANNACH, 2018; GUO *et al.*, 2019; HIDASI *et al.*, 2015):

- **HitRate@N**: measures the capacity of the algorithm to find the immediate next item in a recommendation list of length *N*. HitRate (HR) returns 1 if the next item is present in the list, and 0 otherwise. The final score is given by the ratio between the amount of positive hits and the total of the recommendations given.
- **Mean Reciprocal Ranking (MRR)@N**: shows how high the next item was in the ranked list of recommendations. The metric is calculated as:

$$MRR = \frac{1}{|RL|} \sum_{i=1}^{|RL|} \frac{1}{rank_i}$$

where *RL* is the Recommendation List and $rank_i$ is the position of the next item *i* in the list. The metric returns 0 if *i* is not in *RL*. For example, $MRR@N = 0.5$ indicates that, in average, most accessed recommendations were the top-2 items of the list.

2.3.2 Beyond-accuracy metrics

Relevance can vary depending on users' and companies' interests. The metrics described so far evaluate how accurately a system can identify users' overall preferences, but many factors can influence the degree of satisfaction of the users towards the recommendations they receive. To evaluate these factors, KAMINSKAS; BRIDGE have also proposed the following metrics:

- **Diversity**: which evaluates the diversity of characteristics of items of a recommendation list, because a diverse list can increase the chances of a user to enjoy at least one recommendation;
- **Novelty**: evaluates the ability of RS to recommend items that are unknown to most of the users;

² <https://recsys.acm.org/recsys18/challenge/>

- Serendipity: Also known as "lucky surprise", evaluates the capacity of RS to find items that are both unknown and of interest to a user can increase their experience;
- Coverage@N: The percentage of the items of the database an RS is able to recommend with a Recommendation List of N items. Low coverage is undesirable because it indicates that the users are seeing the same set of items, jeopardizing their experience in a platform. This can also be unfair to content providers whose items are not being recommended by the algorithm, limiting their exposure to the audience.

The main features available for most SBRS datasets are session ids, item ids, and the timestamp of the interaction. Metadata of items is a feature needed to measure Diversity, however, not all datasets make these features available. Moreover, serendipity and novelty are desirable especially for users that have interacted with a platform for a long time, but anonymous users, in average, do not have long enough sessions, as seen in Table 6, in order to these metrics to be effective. Coverage, on the other hand, have been used as a metric in SBRS in order to evaluate the overall diversity of recommendation lists.

2.3.3 Validation

In order to avoid a problem called overfitting, where the model specializes in the given set and fails to give accurate predictions in any other dataset, works in the traditional tasks (D'ADDIO; DOMINGUES; MANZATO, 2017; COSTA; MANZATO; CAMPELLO, 2016) have used k -fold cross-validation technique, where the examples in the dataset are randomly divided in k subsets and, for each experiment, the $k - 1$ sets are separated for training and the remaining set is used for testing. However, in the SBRS task of predicting the next item, the order of the interactions are crucial information, therefore, examples cannot be randomly distributed between training and test sets. Therefore, the method proposed in (HIDASI *et al.*, 2015) and used by later works (LUDEWIG; JANNACH, 2018; GUO *et al.*, 2019; GARG *et al.*, 2019) to validate SBRS models has been to filter out sessions with length 1 and items with less than 5 interactions, delimit the last n days of the dataset for testing and use the remaining as the training set. For large datasets such as RSC15, the dataset has been divided either by n splits (LUDEWIG; JANNACH, 2018) or by using different fractions of the dataset, such as $1/64$, used in (GARG *et al.*, 2019).

2.4 Final Considerations

In this chapter we made a brief overview of the academic developments in the area of Recommender Systems. We discussed strengths and weaknesses of traditional approaches, the tasks RS are designed to do, and evaluation techniques to measure their efficiency. As the focus of academic research in this area transitioned from using explicit feedback to implicit

feedback, the importance of understanding the users' context during their platform became essential to delivering interesting recommendations, so contextual features shifted from auxiliary features to the core of recommendations, as we can see in sequence-aware and session-based recommendations. Yet, a broad understanding of the users context in the moment they interact with the platform benefits not only RS, as they have more information to explore, but also the platforms, as they can also analyze cohorts in order to make business decisions.

RELATED WORK

In this chapter we present the area of Session-Based Recommender Systems, exploring different techniques proposed throughout the years, with the focus on neighborhood-based models proposed for the task of recommending the next item for a given session. We will also present works that motivated the application of TARS techniques into our proposed approaches, focusing on recent works on that applied these techniques on the task recommending the immediate next item for session-based recommendations.

3.1 Session-Based Recommender Systems

In Section 2.2.2 we introduced the intuition behind session-based recommendations through the point-of-view of SARS, which assumes that grouping the interactions into sessions while maintaining their order can improve the ability of the recommender to understand long and short term preferences. However, by including earlier works such as Transaction-Based Recommender Systems (Forsati; Meybodi; Neiat, 2009) as SBRS approaches, WANG; CAO; WANG brings the idea that the order of interactions within the same session is not as important as the order the sessions occurred. Although SBRS and SARS shares the same tasks, SBRS can also use algorithms that uses only co-occurrences of items in a session, such as items purchased in a given transaction. This notion is also reinforced by the original S-KNN model (JANNACH; LUDEWIG, 2017), which is a strong baseline model that does not leverage the order of interactions within the same session. The intuition behind this approach is that understanding the overall interest of the user during a session has a bigger impact than processing the order of interactions in a rigid way.

In session-based recommendations, WANG; CAO; WANG have characterized the data available for this task in five different levels, illustrated in Figure 1:

- Domain level: according to the author, earliest works in SBRS have been developed in

Figure 1 – Hierarchical structure of session data and specific challenges in SBRS. Extracted from (WANG; CAO; WANG, 2019)

transaction-based scenarios, where each session would be represented by the purchases made by the users. Point-of-Interest recommendations, and web browsing were also included in this area. Recent works, however, have been focused in the music and e-commerce domains.

- **Session level:** when available, the system can make use of the history of the users' interactions in order to understand their long-term preferences. Moreover, by grouping the interactions into sessions, the system can also understand heterogeneity of the interests between sessions and use these inter-session models to suggest items for the consecutive session.
- **Item level:** at this level, dependency between items are examined, such as co-occurrences and noisy/irrelevant items.
- **Feature level:** when available, systems can leverage features of the items, such as their categories, tags, brands, promotions, genres, in order to improve item-level relationships.
- **Feature-value level:** the values of the features mentioned in the previous item.

Most works listed as SBRS approaches in (WANG; CAO; WANG, 2019) were focused in session-based recommendations that could be enhanced by inter-session. However, the evaluation of state-of-the-art SBRS algorithms made by (LUDEWIG; JANNACH, 2018) suited the problem of anonymous session-based recommendations because the models considered all past sessions as independent sessions. E-commerce can greatly benefit from this approach because recommending

relevant items to anonymous users can be a good way to convert these users into regular users. Media streaming sessions can take advantage of these scenario, since these approaches can give exclusive focus on the context of the session, and producing playlists of items that occurred in the same context as a result. Moreover, a desirable feature of these platforms is the ability to create infinite playlists, which depends on delivering good next-item recommendations.

Table 2 exemplifies a dataset used for SBRS works. The datasets commonly used in this area comes with ids of sessions already determined, but sessions can also be heuristically defined, using the timestamps of the interactions as base. The timestamp of the interactions is an important feature to determine the sequence of interactions. However, this information can be masked by using a relative timestamp from the first interaction, or simply giving the order of the interactions, in case platforms do not feel comfortable sharing interaction patterns.

Table 2 – Example of a dataset prepared used for SBRS studies.

SessionId	ItemId	Timestamp
s1	i1	0
s1	i2	1
s1	i3	2
s1	i4	3
s2	i4	0
s2	i1	1
s2	i6	2
s3	i3	4
s3	i2	5
s3	i5	6
s4	i1	4
s4	i5	5
s4	i4	6
s4	i2	7
s5	i3	6
s5	i2	7
s5	?	8

In anonymous session-based recommendations, two tasks can be highlighted:

- Predict the immediate next item, given the first n elements: a common task in case of e-commerce, but can also be useful for streaming platforms. An illustration of the scenario can be seen in Table 3.
- Predict all subsequent items: this scenario is more related in media streaming platforms, when the platform recommends a playlist for the ongoing session, refining after every interaction.

The following subsections will briefly describe SBRS techniques used as baselines for SBRS works, focusing on the baselines in the most recent survey (LUDEWIG; JANNACH,

Table 3 – Representations of sessions s_n in SBRS. Sessions are represented by lists of items i_m that users interacted within a given session. In the illustration, the task of the recommender is to predict the next item that u_5 would interacted, characterized by “?”.

s_1	i_1	i_2	i_3	i_4
s_2	i_4	i_1	i_6	
s_3	i_3	i_2	i_5	
s_4	i_1	i_5	i_4	i_2
s_5	i_3	i_2	?	

2018) used to compare the efficiency of different techniques in session-based recommendations that considered sessions as independent group of events, which is the closest scenario to recommending items to anonymous session.

3.1.1 Association Rules

The Association Rules (AR) technique analyzes the co-occurrences of items within a session, groups items into itemsets and recommends items that had the highest frequencies according to the itemset that matches the ongoing session. Since the management of the ideal size of the itemsets can be challenging, LUDEWIG; JANNACH used as baseline an AR with a rule size of 2, which means that this algorithm yields recommendations by selecting the items with highest co-occurrences with the last item of the session, which, in some scenarios, served as a good baseline model for SBRS tasks.

3.1.2 Markov Chains

Markov Chains (MC) approaches are established techniques in sequence-based RS. In the simplified model used in the survey, the algorithm maps the amount of times a set of items have occurred after a given session state, which is characterized by the sequence of interactions. For example, a first-order MC would map the amount of times a set of items was viewed immediately after an item p . The recommendations for an ongoing session would be given based on the most frequent items that were accessed after the last item of the session.

3.1.3 Sequential Rules

LUDEWIG; JANNACH proposed Sequential Rules (SR) as a variation of both AR and MC simplified implementations presented before. In this case, the recommender will also recommend items based on the last item of the session. Given the last item of the session p , the score of an item q is given by a weighted sum of times q appeared after p in previous sessions, where the weight function w is given as $w_{SR}(x) = 1/(x)$, where x is the number of steps between the two items. This approach was among the best performing algorithms for the music datasets, but did not perform as well in the e-commerce ones.

3.1.4 Neural Network models

A statistic of publications on SBRS presented in (WANG; CAO; WANG, 2019) showed that several works have been developed using neural network models in this area. Shallow neural network models such as Item2vec (Barkan; Koenigstein, 2016) adapted word2vec models by considering words as items and sessions as sentences in order to create item embeddings that could be used to find item analogies. Recurrent neural network models with Gated Recurrent Units have also been explored for modeling long-term dependencies within a sequence. These models gained popularity with GRU4Rec (HIDASI *et al.*, 2016), and have received extensions such as attention mechanisms to capture the main intentions of the current session (LI *et al.*, 2017a). On the other hand, (QUADRANA *et al.*, 2017) have created hierarchical neural models to better capture inter-session information.

These neural network models have been successfully used in scenarios where inter-session information are available. (LUDEWIG; JANNACH, 2018) has shown that they are also strong baselines for anonymous sessions in large datasets. However, due to the high cost of the training stage of these models, as well as hyperparameter tuning, many experiments have been published using only a fraction of these large datasets. Moreover, (MI; FALTINGS, 2018) argued that non-parametric methods such as the S-KNN model can consistently outperform these models in scenarios where new items are constantly added to the system.

3.1.5 Factorization models

Works using factorization models have adapted the traditional model by incorporating the session and sequential information into the models. In (LUDEWIG; JANNACH, 2018) the authors compared different adaptations of MF techniques found in the literature for session-aware recommendations with the MF technique proposed in the paper. Most of the models combined factorized Markov chains into the classic matrix factorization in order to model the sequence of the interactions. In the rating function of these models, in addition to the traditional $p_u \times q_i$ calculations, a sequential dynamics weight $n_l \times m_i^T$ is added to the model, where n and m are the latent vector representations for items l and i respectively. However, in most of the datasets used in the experiments in (LUDEWIG; JANNACH, 2018), these factorization models had the poorest performance among the baseline models, having the Session-based Matrix Factorization technique, which was specifically designed for the next-item recommendation, the best of this class of algorithms.

3.1.6 Neighborhood-based Models

The adaptations to the traditional neighborhood-based models mentioned in Section 2.1.1.1 to SBRS are described below:

The Item-based KNN (I-KNN) recommends the k most similar items to the last item the

session has interacted. In this model, each item is encoded as a vector where the values are the amount of times each session accessed them. The similarity between two items can be determined with metrics such as Jaccard index or cosine similarity, and the item-item similarity matrix can be calculated in the training stage and stored, speeding up the recommendation process.

The traditional I-KNN model takes advantages of the co-occurrence of items in past sessions, but only uses the last interactions when recommending for the current session, which limits the algorithms ability to personalize content as the session progresses.

On the other hand, the algorithm based on a user-based KNN adaptation proposed in (JANNACH; LUDEWIG, 2017), also known as the S-KNN model, transforms the logs of interactions into lists of items grouped by sessions. Figure 2 illustrates the architecture of the S-KNN model, which is split into three stages:

Figure 2 – The architecture of the S-KNN model, from the training stage to the recommendation list to an anonymous user.

- **Pre-processing:** In this stage, the model creates data structures to map all interactions to quickly retrieve information. In this step, the proposed algorithm creates one structure that maps sessions that interacted with the items of the dataset, another one that maps all the sessions with the timestamp of their last interaction, and a third structure that maps the items of the sessions.
- **Sampling Sessions:** As a session is occurring, the model creates a list of relevant sessions (RL) composed of sessions that also interacted with at least one of the items of the

ongoing session. With this, the algorithm reduces the search space when looking for nearest neighbors by guaranteeing that all similarity calculations yields values greater than 0. However, the remaining set can still be very large, so a sampling strategy is applied to return a sample set of m possible neighbors.

- Find items of interest: In this stage, the algorithm finds the k nearest neighbors N_s from RL and calculates the score of the items available for recommendation. The score of an item i to a session s is given by:

$$score_{S-KNN}(i, s) = \sum_{n \in N_s} sim(s, n) \times 1_n(i) \quad (3.1)$$

where $1_n(i)$ is a function that returns 1 if the neighbor session n accessed item i and 0 otherwise.

In order to reduce the search space when finding the nearest neighbors, the model uses sampling strategies to alleviate scalability problems that can occur as the number of sessions increases. From the data structures created from the logs of sessions, the model first create a set of relevant sessions list RL , which are composed of sessions that accessed the same items of the ongoing session.

The authors proposed different strategies to sample sessions from this set of relevant sessions. They proposed using a user-defined parameter m as the number of sessions to be sampled with one of the following strategies:

- Random Sampling: select m sessions from RL randomly;
- Sampling Most Recent Sessions select the most m recent sessions that happened before the time of the recommendation:

Table 4 – On the left, representations of sessions s_n in SBRS. Sessions are represented by lists of items i_m that users interacted within a given session. In the illustration, the task of the recommender is to predict the next item that u_5 would interacted, characterized by “?”. Also, the table on the right represents a structure that can easily retrieve the timestamp of the last interactions of the sessions.

s_1	i_1	i_2	i_3	i_4
s_2	i_4	i_1	i_6	
s_3	i_3	i_2	i_5	
s_4	i_1	i_5	i_4	i_2
s_5	i_3	i_2	?	

i1	s1	s2	s4
i2	s1	s3	s4
i3	s1	s3	
i4	s1	s2	s4
i5	s3	s4	
i6	s2		

SessionId	Timestamp
s1	3
s2	2
s3	6
s4	7
s5	8

We illustrate the procedure of generating recommendations using the example of logs in Table 2, using $m = 3$ and choosing $k = 2$ as the amount of nearest neighbors. The original

S-KNN model proposed in (JANNACH; LUDEWIG, 2017) would create index structures in Table 4 in the pre-processing stage, and put sessions s_1 , s_2 , s_3 and s_4 in the relevant list of sessions RL, as these sessions have interacted with i_2 or i_3 . Since the length of sessions is greater than m , the model samples 3 of the 4 sessions of the list. Using the most recent sessions strategy, the algorithms would select sessions s_1 , s_3 and s_4 . Now, using an adjusted cosine similarity to calculate the similarity between the sampled sessions and s_5 , given by

$$\text{sim}(s_x, s_y) = \cos(s_x, s_y) = \frac{s_x \cap s_y}{||s_x|| \cdot ||s_y||}$$

$$\text{sim}(s_5, s_1) = \frac{2}{\text{sqrt}(2) \cdot \text{sqrt}(4)} = 0.707$$

$$\text{sim}(s_5, s_3) = \frac{2}{\text{sqrt}(2) \cdot \text{sqrt}(3)} = 0.916$$

$$\text{sim}(s_5, s_4) = \frac{1}{\text{sqrt}(2) \cdot \text{sqrt}(4)} = 0.353$$

Therefore, s_1 and s_3 are the k most similar sessions to s_5 . The score of each item, thus, would be calculated as:

$$\text{score}_{SKNN}(i_1, s_5) = 0.707$$

$$\text{score}_{SKNN}(i_2, s_5) = 0.707 + 0.916 = 1.623$$

$$\text{score}_{SKNN}(i_3, s_5) = 0.707 + 0.916 = 1.623$$

$$\text{score}_{SKNN}(i_4, s_5) = 0.707$$

$$\text{score}_{SKNN}(i_5, s_5) = 0.916$$

If the recommendation list was made of the top-3 relevant items, the list or recommended items for s_5 , in this example, would be formed by i_2 , i_3 , and i_5 , where the order between i_2 and i_3 could be chosen randomly for in this algorithm.

(JANNACH; LUDEWIG, 2017) argued that sampling the most recent sessions can improve recommendations especially in e-commerce datasets, indicating that sessions follow seasonal tendencies in this domain, whereas sampling at random led to only small accuracy compromises, which made the S-KNN model a strong baseline for SBRS tasks.

3.1.6.1 S-KNN extensions

S-KNN extensions have been proposed using time-related contexts in various ways. LUDEWIG; JANNACH also proposed sequence-aware extensions for their initial model, since it not leveraged the order of interactions in the recommendation process. The extensions proposed were:

- Sequential Session-based KNN (S-SKNN): This variant changes the score function of an item by assigning a weighting function, for example, $w_n(i, s) = x/|s|$, where x is the position of the latest item i that co-occurred in session s and a neighbor session n .

$$score_{SKNN}(i, s) = \sum_{n \in N_s} sim(s, n) \times w_n(i, s)$$

The weighting function guarantees that items accessed later in the session also have more importance in the recommending process.

In the previous example, the score for the items i_4 and i_5 for s_5 would be given by:

$$score_{S-SKNN}(i_1, s_5) = 0.707 \times 1/4 = 0.177$$

$$score_{S-SKNN}(i_4, s_5) = 0.707 \times 4/4 = 0.707$$

As we can see, S-SKNN could give different scores to i_1 and i_4 by lowering the score of i_1 since it was accessed earlier in the session s_1 compared to i_4 .

- Sequential Filter Session-based KNN (SF-SKNN): This post-filtering method modifies the scoring function by adding an *happened_before* function that returns 1 in case last item j has ever been accessed before the item-to-be-recommended i . Therefore, the original scoring function (shown in equation 3.1), becomes:

$$score_{SF-SKNN}(i, s) = \sum_{n \in N_s} sim(s, n) \times 1_n(i) \times happened_before(j, i) \quad (3.2)$$

Back in the example, only i_3 and i_5 would have scores greater than 0, since they are the only items where i_2 has been interacted before them.

- **Vector Multiplication Session-Based KNN (V-SKNN):** This model transforms a binary representation of the ongoing session into a representation that puts more emphasis on the latest events of a session by only assigning "1" to the last accessed item and using a decay function to previous items. The similarity is calculated with a dot product between the current weight-encoded session and binary-encoded past sessions.

In the example, using the same weight function used in the S-SKNN model to generate the vector representations, the similarity between s_5 and s_1 would be given by:

$$\text{sim}(s_5, s_1) = \frac{1 \times 1/2 + 1/2 \times 3/4}{\text{sqrt}(2) \cdot \text{sqrt}(4)} = 0.486$$

However, [GUO et al.](#) pointed out that sampling the m most recent sessions, although effective, does not guarantee that all of the items accessed by the ongoing session had at least one session in the set to be considered in the final set of candidate sessions. For example, if another session s_6 appeared interacting with items i_2 , i_5 and i_6 , having the size of the sampled list $m = 3$ would bring sessions s_5 , s_4 and s_3 using the most recent sessions sampling strategy. This would leave s_2 out of the list, even though it was the only session that also interacted with i_6 . In order to solve this problem, the authors proposed a sampling strategy called Equal Probability Candidate Selection at Random (EPCSR), which, given a set of relevant sessions RC, a set of items of the ongoing session x , and sample size m , the sampling strategy is:

- **Last Interaction:** The model will separate a ratio of $p_i = \frac{m}{|x|}$ for the most recent sessions of the last interactions, as the last interaction is considered the most relevant interaction;
- **Other interactions:** The other interactions will be sampled at random, respecting the remaining of $1 - p_i$ ratio for each interaction.

With this strategy, s_2 would be included in the relevant list of sessions for s_6 , and the other 2 sessions would be selected at random. The model also proposed a Diffusion Similarity Measure between two sessions x and j , which is generated by considering the interactions as a bipartite session-item network, calculated as:

$$\text{Sim}_{DSM}(x, j, \lambda, \beta) = \frac{1}{d_x^\lambda \times d_j^{1-\lambda}} \sum_{i=0}^n \frac{a_{xi} \cdot a_{ji}}{d_i^\beta} \quad (3.3)$$

where n is the size of the intersection set of items between x and j , λ is a parameter that adjusts the influence of the degree of the sessions and β is another parameter that adjusts the degree of the items i . In this case, the degree of an item is equivalent to its popularity, and the degree of a session is equivalent to the size of the session. Therefore, setting $\lambda = 0.5$ and $\beta = 0$ would be equivalent to using a cosine similarity.

The Sequence and Time-Aware Neighborhood (STAN) extension (GARG *et al.*, 2019), on the other hand, used the timestamp of the interactions to create three different weight functions for the model. The first weight function w_1 modifies the similarity function by leveraging latest accessed items of the ongoing session. w_1 is given by:

$$w_1(i, s) = \exp\left(\frac{p(i, s) - |s|}{\lambda_1}\right) \quad (3.4)$$

which is applied to each item i of a session s . $p(i, s)$ is the position of i in s , and λ_1 is a user-defined parameter. This transforms the binary representation of a session into a real-valued vector s_w , which is later used to calculate the similarity between s_w and a past session s_j , given by:

$$\text{sim}_1(s, s_j) = \frac{s_w \cdot s_j}{\sqrt{|s| \cdot |s_j|}} \quad (3.5)$$

This first step is equivalent to the approach given by V-SKNN. However, the similarity between s and s_j receives another weight function w_2 that penalizes the score the further s_j happened compared to s . Given $t(s_j)$ the timestamp of the last item accessed by s_j , $t(s)$ the timestamp of the last item accessed by s , and λ_2 another user-defined parameter to control the influence of this weighting function, w_2 is given by:

$$w_2(s_j | s) = \exp\left(\frac{t(s_j) - t(s)}{\lambda_2}\right) \quad (3.6)$$

Finally, the similarity between an ongoing session s and past session s_j is given by:

$$\text{sim}_2(s, s_j) = \text{sim}_1(s, s_j) \times w_2(s_j | s) \quad (3.7)$$

In the scoring function, there is another weight function w_3 that ponders the distance between an item j that occurred both in the session s and a neighbor session n . w_3 is given by:

$$w_3(i | s, n) = \exp\left(-\frac{|p(i, n) - p(j, n)|}{\lambda_3}\right) \times 1_n(1) \quad (3.8)$$

where $p(i, n)$ and $p(j, n)$ are the positions where i and j occurred in n , respectively, and λ_3 is another user-defined parameter.

With this, the score STAN gives to an item-to-be-recommended i to a session s is given by:

$$\text{score}_{\text{STAN}}(i, s) = \sum_{n \in N(s)} \text{sim}_2(s, n) \times w_3(i | s, n) \quad (3.9)$$

STAN incorporates context in three different stages of the recommendation, making use of the order of the interactions both in w_1 and w_3 , and uses w_2 to forget older sessions. However,

working with real-valued vectors instead of binary vectors increases the calculation time for the recommendations.

3.2 Applications of TARS approaches in Session-Based Recommendations

In (TAN; XU; LIU, 2016), the authors leveraged more recent subsets of the dataset by first training a Gated Recurrent Unit RNN model using the entire dataset, and using the trained model as the initialization of a new model that will be fitted using only a smaller recent subset of the data. Among the strategies proposed, this extension of the RNN model yielded the best results in the experiments. BOGINA; KUFLIK also extended the GRU4Rec model by changing a sampling strategy used in the training stage of the RNN model by multiplying the number of instances of the items according to their dwell times. If an item surpasses a pre-defined threshold for a dwell time t by x times, then the item will be represented x times in the sequence. Besides the improvement in the accuracy over the baseline models, another contribution made by the authors was the statistical analysis about the dwell times between items. Although RNN models have been largely explored in SBRS in general due to their ability to accurately interpret the sequence of actions of the users, they often need large amounts of data and preferably long user histories in order to perform well. However, working with historical data of the users is not possible for anonymous sessions, and cold-start scenarios, considering the new item problem and new user problem (MI; FALTINGS, 2018), negatively affect the performance of these models. A different usage of TARS is seen in (STEFANCOVA; SRBA, 2020), where the authors used a factorization model with a periodic-based pre-filtering approach, dividing the interactions into months of the year with a location-based technique to recommend places in a Point-of-Interest (POI) system. They also used a post-filtering approach by raising the scores of items that were nearby the user. Their successful usage of the pre-filtering approach indicated that seasonality is an interesting feature to be explored in this domain, and periodic-based approaches can be successful in SBRS. In POI recommendations, the next action of the user is to go the place the user intended to go, therefore, a much more expensive action than Web searching, listening to music or viewing products on an e-commerce. Therefore, these algorithms have to rely on historic information about the user.

3.2.1 TARS approaches with Neighborhood-based models

Among the works using time-aware techniques in SBRS, the original S-KNN model uses a recency-based approach by sampling the most recent sessions in order to reduce the search space. This approach, alongside the reported the results of the number of days used for training and performance of the model, suggested that recency trends can be an important feature to be explored in SBRS, especially in the e-commerce domain (LUDEWIG; JANNACH, 2018).

(GUO *et al.*, 2019) improved this sampling strategy by guaranteeing that all items of the current sessions had representative sessions in the sampling set of sessions. These results were important indicators that more investigation should be done to elaborate effective sampling strategies for the S-KNN model. STAN(GARG *et al.*, 2019) have also applied a recency-based approach in the model by applying a weighting function to lower similarity scores to sessions that occurred further in time. However, the additional step in the similarity calculations added significant computational costs to the model. Besides, the model is dependent to the optimal parameters used to determine the importance of the weighting factor in the model.

Since S-KNN model does not leverage the order of interactions in the recommendation process, (LUDEWIG; JANNACH, 2018) and (GARG *et al.*, 2019) have also proposed sequence-aware extensions to the original S-KNN model. The V-SKNN model and the first weighting factor in STAN transforms the binary vector representations of sessions into a real-valued vectors to be used in the similarity functions when finding the nearest neighbors. Since working with real-valued vectors increases computational costs compared to binary representations, having an effective sampling strategy can greatly benefit the computational cost of these models. (GUO *et al.*, 2019), on the other hand, have raised the importance of the popularity of the items in the model by considering the degree of the item in the similarity function proposed, therefore using a different contextual factor in the model.

Finally, extensions to the model were also proposed in the scoring function of the items. The S-SKNN model (LUDEWIG; JANNACH, 2018) changed the scoring function by applying more weight to the latest accessed items of the sessions. The SF-SKNN model, proposed by the same authors, applied a filtering function that would not consider items that were accessed after the last item of the session, an approach more inspired by Markovian models. Since these models are applied only in the last step of the recommendation process, these approaches can also benefit from effective sampling strategies. The STAN model also proposed a weighting function that assigns lower weights relatively to the position of the items that were accessed earlier in the session, however, this weighting function yielded the poorest performance compared to the other weighting functions (GARG *et al.*, 2019).

3.3 Final Considerations

In session-based recommendations, most works developed have focused on the task of finding the next-item of interest. However, these works also assumed that these recommendations would be given to users that had multiple interactions recorded in the system, so that inter-session information could be encoded in these models to improve recommendations to the current session.

As pointed out by RUOCCO; SKREDE; LANGSETH, SBRS that only consider the current session has to recommend items using only a small set of interactions. This is the default

case for anonymous users. Although deep learning techniques became popular in SBRS (GUO *et al.*, 2019), neighborhood-based models have been able to outperform these models in the session-based recommendations when previous sessions are not considered (LUDEWIG; JANNACH, 2018), while being simpler models and being able to produce easy-to-explain recommendations. Moreover, in (STEFANCOVA; SRBA, 2020; BOGINA; KUFLIK, 2017) the exploration of the datasets used according to the context they captured helped the explanation of the performances of the algorithms.

Among the different approaches presented in this chapter, neighborhood-based models in session-based recommendation tasks have motivated the development of this work due to their success in scenarios closely related to anonymous sessions recommendations, while being a simple model compared to state-of-the-art MF and Neural Network models. Moreover, MI; FALTINGS also argued that non-parametric models such as the KNN models also adapts better than the mentioned models in cold-start scenarios for items. Although traditional KNN approaches can have performance limitations, the S-KNN model successfully addresses their main concerns, but still leaves opportunities for improvement.

The original S-KNN model uses the timestamp of the interactions in order to filter the most recent sessions from a set of candidate neighbors, working as a pre-filtering CARS strategy. While the extensions proposed in (LUDEWIG; JANNACH, 2018) also used the timestamps in order to retrieve the sequence of interactions, GUO *et al.* proposed an adjustment in the selection of possible neighbors that led to significant improvements, which motivated this work to follow this path of work and propose a different pre-filtering strategy. GARG *et al.*, on the other hand, have explored the relative distances between items in the sequences of interactions, as well as incorporating time into the similarity calculations. Finally, the SF-SKNN model is an example of a post-filtering technique that leverages the sequence between items in sessions. All these techniques have used timestamps in an interactional view, but none of them have explored this feature as a representational view, that not only has the potential of improving traditional recommenders (BALTRUNAS; AMATRIAIN, 2009; STEFANCOVA; SRBA, 2020), but also yield other analysis related to how characteristics of the dataset can influence the performance of the algorithms.

The next chapter contains the proposal of this project, divided in objective and proposed extensions to the S-KNN model, as well as the contexts used to test these extensions.

PROPOSAL

Previous chapters presented fundamental concepts and techniques about Recommender Systems that served as the basis of this work. In this chapter, we present our proposal, which is divided in Project Justification in Section 4.1, where we reinforce the main points that guided the development of this work, Section 4.2 describes our proposed models, in Section 4.3 we present the context extracted for our approaches, and we close the chapter with final considerations in 4.4.

4.1 Project Justification

Timestamps of the interactions are data that can be easily captured and can give insight about behavior of users in different contexts, such as the period of the day, day of the week which most sessions happened and which items had more interactions. With this information in hand, platforms are able to develop different marketing strategies to improve users' and businesses' engagements. In session-based recommendations, timestamps allow techniques to leverage the sequence of interactions as a feature to infer users' intent. Moreover, contexts inferred by this feature does not need any additional information from the platforms, so no additional privacy concerns are raised.

The capacity of producing accurate recommendations while being easy-to-implement and easy-to-explain are desirable qualities of a recommender system. The S-KNN model and its extensions are neighborhood-based techniques that operate with a "people who consume this also consume that" logic when making recommendations. These models have proven to be strong baselines in SBRS tasks, outperforming deep neural network models in the task of recommending items to anonymous sessions (LUDEWIG; JANNACH, 2018). However, for anonymous sessions, systems have to adjust their recommendations for anonymous sessions during an ongoing session, and this can be a limiting factor in neighborhood models, since most of the processing is done in the recommendation process. These models have known computational and space costs since

these constraints grow exponentially as the amount of data grow, so these models benefit greatly with mechanisms that reduce the search space when finding the nearest neighbors of the ongoing session.

Works that explored neighborhood-based models (LUDEWIG; JANNACH, 2018; GUO *et al.*, 2019) in SBRS tasks indicate that sampling strategies to reduce the search space of possible neighbors can soften these problems without compromising the accuracy of the recommendations. By using recency of the interactions as a context to be used as a pre-filtering approach, these works have proven that time-related contexts are useful features to be considered in this scenario. context was used to improve the scoring function by leveraging either the scoring function (S-SKNN and SF-SKNN) or the similarity function (V-SKNN).

Therefore, this work explores the efficacy of incorporating easy-to-extract time-related contexts into S-KNN models in order to improve the quality of the recommendations. With this, we intend to show how meaningful contexts that are easily extracted from the datasets can impact these state-of-the-art algorithms in SBRS.

4.2 Proposed Models

In this section we describe the models proposed to evaluate the impact of extending neighborhood-based models with periodic-based approaches in SBRS. The first architecture extends the S-KNN model by using a pre-filtering strategy, that sample sessions according to the context of the session. The underlying hypothesis for this approach is that sessions that happened in the same context have inherently more relevance, and, by using context as a filter, we can reduce the amount of times the sorting of the most m recent sessions is needed to reduce computational costs. The second architecture uses context as a post-filtering strategy so that the lists of recommendations previously generated by the original S-KNN can be improved with contextual information. Pre-filtering approaches have the possibility of filtering out too many examples, making the amount of information left insufficient to provide recommendations. However, works such as (BALTRUNAS; AMATRIAIN, 2009; STEFANCOVA; SRBA, 2020) have indicated that incorporating these time-related information can improve recommendations while also yielding information about interactions that can be explored in different analysis, such as the performance of the algorithms with different views of contexts.

4.2.1 Context as a pre-filtering strategy

The first architecture, illustrated in Figure 3, adds another data structure to map all sessions by the context they occurred, so that the model can filter from the relevant set of sessions all the examples that did not occur in the same context.

Resuming the example of Table 2, this approach includes creating an additional data structure that stores sessions according to the context they happened, in order to quickly retrieve

Figure 3 – Illustration of the first proposed extension for the S-KNN model. The content inside the blue box represents the additional data structure created to map sessions by context, which is used by the sampling step to filter out sessions that did not happen in the same context. The contexts used are described in section 4.3.

them in the recommendation process (Table 5 illustrates the additional structure from the example).

Table 5 – Contexts extracted from the example log.

SessionId	Context
s1	0
s2	0
s3	1
s4	1
s5	0

The initial list of relevant sessions is composed of sessions s_1 , s_3 and s_4 . if the sample size is $m = 2$, we could apply the pre-filtering strategy in order to filter the sessions that happened at context 0, which would filter sessions s_3 and s_4 , leaving only s_1 to recommend. As seen in the example, This approach, as well as any window-based approach in TARS, have the concern of filtering too many examples, leaving a rather small amount of examples to be used in the recommendation process. However, if the context used to filter sessions is actually relevant

in online scenarios, this sampling strategy could still yield accurate recommendations while reducing the search space.

4.2.2 Context as a post-filtering strategy

The second architecture, illustrated in Figure 4, focuses on using the data structure designed to map sessions per contexts, mentioned in the previous model, to leverage the co-occurrence of sessions in the same context. That can be achieved by modifying the scoring method of items, and the modifications to the scoring functions are described in section 4.2.3

Figure 4 – Illustration of the second proposed extension for the S-KNN model. The extension proposed in the architecture is highlighted by the blue box, where the proposed scoring methods uses the indexing of sessions per context to lower computational costs. The contexts used are described in section 4.3.

This model, as the previous one, would also create a structure that maps sessions and their contexts.

4.2.3 Scoring methods

In order to leverage the importance of the sessions that co-occurred in the same context, we can raise the importance of these sessions by modifying the scoring method. In this work, we

proposed two different modifications:

- **Context-Filter S-KNN (CF-SKNN):** Modifies the original S-KNN scoring method by only calculating scores for items of neighbors occurred in the same context of the current session. This works as a post-filtering strategy, filtering out from the nearest neighbor sessions N_s all sessions that did not happen in the same context. Given an item-to-be-recommended i , the ongoing session s , a neighbor session n , the scoring function is:

$$score_{CF-SKNN}(i, s) = \sum_{n \in N_s} sim(s, n) \times 1_n(i) \times same_context(s, n) \quad (4.1)$$

Where $sim(s, n)$ is a similarity function to be used, $1_n(i)$ a function that returns 1 if i was accessed by n and 0 otherwise, and $same_context(s, n)$ is another function that returns 1 if s and n happen in the same context, and 0 if they do not. One limitation of this approach is that, if no sessions in N_s were in the same context of s , no scores will be given to items. When this situation happens, items will be recommended at random.

In the example for the original S-KNN model, if using $m = 3$ and $k = 3$, the model would use sessions s_1 , s_3 and s_4 to calculate the score of items. However, since we are using a context filter, items accessed by s_3 and s_4 would not have their scores raised.

- **Inverse-Context-Frequency S-KNN (ICF-SKNN):** This approach intends to model contexts into the original S-KNN scoring method by raising scores for items of neighbors occurred in the same context c of the current session. Given an item-to-be-recommended i , the ongoing session s , a neighbor session n , the scoring function is:

$$score_{ICF-SKNN}(i, s) = \sum_{n \in N_s} sim(s, n) \times 1_n(i) \times w(n, c) \quad (4.2)$$

where the weighting function $w(n, c) = \sqrt{\frac{|N_s|}{|N_s| \cap |c|}}$ if $(|N_s| \cap |c|) \neq \emptyset$, otherwise 1.

This weighting function intends to make the items of neighbors that co-occurred in a context more relevant, raising the relevance of the context the rarer it appears in N_s . Unlike the previous approach, ICF-SKNN will act as the original S-KNN in case no sessions in N_s co-occurred in the given context c , but will raise the importance of co-occurring sessions in case there are any. The square root function acts as a way to soften the importance of the context in case the co-occurrences are rare in N_s .

So, resuming the example, with $m = 3$ and $k = 2$, the score for item i_1 would be given as:

$$score_{S-SKNN}(i_1, s) = 0.707 \times \sqrt{2/1} + 0.916 = 2.33$$

4.3 Context Extraction

When analyzing the amount of interactions by hour and days of the week, platforms can assume that interactions with the system will not be made in a balanced way along the hours of the day and the days of the week, so working with contexts based on perceived separations of time, such as dividing days of the week into workdays (monday through friday) and weekends (saturday and sunday), or morning and evening help algorithms to find examples for contexts with low amount of interactions. However, even these perceptions can vary depending on the geolocation and culture. For example, the weekend can start after work hours on a friday, and, if considering day and night according to the hours the sun rises and sets, these notions can vary even between seasons of the year. Therefore, the heuristics used to extract the contexts used in this project were:

- The day of the week the interactions happened, which is referenced as *weekday* in our analysis;
- The hour of the day the interactions happened, which is referenced as *hour* in our analysis;
- The period of the day the interactions happened, which is referenced as *period* in our analysis. For each period of the day, we considered:
 - Morning: from 6:00 AM to 12:59 PM;
 - Afternoon: from 1:00 PM to 6:59 PM;
 - Night: from 7:00 PM to 11:59 PM;
 - Dawn: from 12:00 AM to 6:00 AM;
- Workday/Weekend: separates the dataset into workdays (Monday through Friday) and weekends (Saturday and Sunday). This view of context is referenced as *workday/weekend* in our analysis;
- Even/Odd hours: separates the interactions if they happened in an even hour ,such as 2:00PM, or odd hour, i.g. 1:00PM. We used this context as a way to check if a meaningful separation of the dataset through contexts would be more relevant than a random yet balanced set of contexts, similar to the analysis presented in (BALTRUNAS; AMATRIAIN, 2009). This view of context is referenced as *even/odd hours* in our analysis;
- Day/Night: Considers day the period between 6 AM and 6 PM, night otherwise. This view of context is referenced as *day/night* in our analysis.

4.4 Final Considerations

In this chapter we presented the motivations that served as basis of this work, as well as the proposed extensions in the S-KNN model in order to analyze the effects of using temporal

contexts as features for neighborhood-based models. We proposed two different architectures, in which we were able to propose an extension as an additional pre-filtering strategy to the original model, a scoring method that acts as a post-filtering approach, and one that incorporates the rarity of the contexts into the original model. One advantage of these extensions, mainly the pre-filtering one, is that they can be easily combined into the other extensions, which allows us to evaluate the importance of each context elaborated for this work in different settings. In the next chapter we will present the results of this work, showing how the application of these temporal contexts affected the algorithms and how they compare to each other.

EVALUATION

This chapter contains the methodology used, as well as the results and discussion of the experiments made for this project. We divided the sets of experiments into two parts: the first set focuses on the analysis of the performance of the first proposed model, while second set of experiments focuses the analysis of the performance of the second architecture, which applied contextual information in the scoring methods of the model.

5.1 Methodology

This section describes the datasets used, evaluation protocol, as well as the tools used to evaluate our proposal.

5.1.1 Datasets

We used publicly available databases that have been commonly used for SBRS works (HIDASI *et al.*, 2016; LUDEWIG; JANNACH, 2018; GUO *et al.*, 2019), analyzing the datasets through the contexts proposed in Section 4.3 as we explored the data. We chose three datasets where the timestamp of the interactions was available, which were:

- Retailrocket: The company RetailRocket published this e-commerce dataset¹ of user activities in a competition hosted at Kaggle.
- RSC15: This e-commerce dataset was published for the ACM RecSys 2015 Challenge by Yoochoose, containing six months of logs of click interactions. This dataset is the established dataset for SBRS work, having earliest explorations made in (HIDASI *et al.*, 2016).

¹ <https://www.kaggle.com/retailrocket/ecommerce-dataset>

- Nowplaying: This is a music dataset created from music-related tweets, where users posted which tracks they were currently listening to (ZANGERLE *et al.*, 2014).

Although the work made by (LUDEWIG; JANNACH, 2018) showed more datasets that can be used to evaluate SBRs tasks, not all of them provided the original timestamp of the interactions. Datasets such as TMall² and diginetica³, for example, are also used in SBRs works and competitions. Since this feature was essential information for the development of this work, they were not considered.

Table 6 presents statistics of the datasets mentioned above, after applying filters that removed sessions of length 1 and items with less than 5 interactions, protocol used in (LUDEWIG; JANNACH, 2018; HIDASI *et al.*, 2016). The statistics presented for Retailrocket and Now playing showed similarities in number of actions and items, and average number of interactions per item. However, the average number of interactions per session is more than doubled, compared to the e-commerce dataset. Although the number of actions in the RSC15 dataset being almost 20 times as large as the previous datasets, the average session length is almost the same as the other e-commerce dataset.

We also decided to analyze the long tail distribution of interactions of the datasets. The Pareto's Principle⁴ is a concept that describes a phenomenon of concentration of sales. That indicates that a small portion of the items are responsible for the majority of sales. The principle is also known as the 80/20 rule, and, in a SBRs scenario, would mean that the a ratio such as the top 20% of the items would be responsible for 80% of the interactions. Some works (BRYNJOLFSSON; HU; SIMESTER, 2011) have explored the impact of RS in the consumption of items in the long-tail distribution of sales, indicating that this phenomenon still persists, even with mechanisms such as RS enabling users to find more niche products, which raises the consumption of the items in the long tail.

For anonymous sessions, the platforms often offer impersonal recommendations, such as the top-n most viewed products, in order to introduce users to their catalog of products. After the initial interaction, RS can start inferring the users' short term interests, their experience to the platform. We calculated the importance of the most 20% popular items in order to verify the skewness of the consumption of items. We verified that for Retailrocket and Nowplaying, this portion of the items generated around 62% of the interactions. However, for RSC15, the top 20% most viewed items represented 90.4% of the interactions.

When exploring the database through the contexts proposed, more differences between datasets were exposed. When analyzing the interactions per days of the week, Figure 6 shows that Retailrocket's accesses get lower toward the weekends, whereas RSC15 interactions peaks

² <https://tianchi.aliyun.com/dataset/dataDetail?dataId=53>

³ <https://competitions.codalab.org/competitions/11161>

⁴ <https://www.interaction-design.org/literature/article/the-pareto-principle-and-how-to-be-more-effective>

Table 6 – Statistics of datasets RetailRocket, RSC15 and Nowplaying after filtering sessions with at least 2 interactions and items with at least 5 interactions.

Dataset	RetailRocket	RSC15	Nowplaying
Actions	1,460,092	31,713,447	1,251,654
Sessions	366,425	7,986,524	150,980
Items	60,627	37,486	61,221
Actions per Session	3.98	3.97	8.29
Actions per Item	24.1	846.0	20.4
% of top 20% popular items	62.1	90.4	62.7

on Sundays and Mondays. Nowplaying, on the other hand, shows an even distribution among the days of the week.

Figure 5 – Number of interactions by hour for the datasets RetailRocket, RSC15 and Nowplaying, respectively.

Figure 5 shows almost opposite patterns of interactions for the e-commerce datasets: in Retailrocket, most interactions happen between 6:00 AM until 8:00 PM, whereas RSC15 has very low interactions between 7:00 AM and 2:00 PM, but peaks at 8:00 PM. The Nowplaying dataset has a less imbalanced distribution of interactions throughout the days. Similarly to the e-commerce datasets, the peak access hour of this dataset is around 6 PM and 8 PM.

Figure 6 – Number of access by day of the week for datasets Retailrocket, RSC15 and Nowplaying.

When separating the data by weekday, both Retailrocket and Nowplaying have less imbalanced distribution of the interactions between the days. RSC15, however, has peaks of access between Sunday and Monday, and a lower audience on Tuesdays, whose accesses are more than three times lower than Sundays.

5.1.2 Evaluation

We adapted the sliding-window protocol used in (LUDEWIG; JANNACH, 2018) for RSC15, dividing the six months of data available into 6 slices of data of 28 days separated for training and 2 days separated for testing. For both Nowplaying and Retailrocket datasets, however, we followed the single training-test split used in (HIDASI *et al.*, 2016), using the last 7 days of interactions for testing, and the remaining for training. This adaptation was made so that the evaluation of the “day of the week” context would not be made upon just one of the weekdays. Moreover, even when using a single split, the amount of average neighbors for the *weekday* and *hour* contexts were very low, below the k parameter that limits the amount of neighbors used for recommendations, as observed in Table 7.

5.1.3 Metrics

In order to measure the performance of the algorithms, we chose established metrics $HR@N$, $MRR@N$ and $Coverage@N$ described in Section 5.1.2 with $N = 20$, as used by previous works (HIDASI *et al.*, 2016; LUDEWIG; JANNACH, 2018; GUO *et al.*, 2019). $HR@20$ measures how well the algorithms are able to propose items of interests by checking if the next items of past sessions in given states were in the list proposed by the algorithm. $MRR@20$ complements $HR@20$ by penalizing scores when the predicted item is not in the top of the list, which also reduces the need to evaluate different values for N . $Coverage@20$ is commonly used to check the diversity of the items in the lists of recommendations, and was also used in this work to check beyond-accuracy metrics.

5.1.4 Auxiliary Resources

This work was developed using the codebase made available in LUDEWIG; JANNACH for baselines, dataset splitting and evaluation algorithms. However, some refactoring were made so that the base S-KNN model algorithm could be inherited by its extensions, such as the SS-KNN and the algorithm presented in (GUO *et al.*, 2019). This codebase was developed using Python 3 and popular libraries such as NumPy and Pandas.

5.2 Experiments

5.2.1 Using context to sample sessions

In the work-in-progress (DOMINGOS *et al.*, 2019), we explored the use of the contexts as a sampling strategy of sessions in order to reduce the search space to reduce the cost of finding the k nearest neighbors. In the paper, we reported that using the contexts proposed to filter sessions lead to worse results, suggesting that the lack of enough relevant sessions to use in the recommendation process was one of the reasons. For example, Figure 5 shows that the

amount of interactions made between 6 AM and 9 AM are very scarce in comparison to other contexts, which can limit the efficiency of the algorithm to find items based on sessions in the same context.

To further investigate the influence of the contexts used as sampling strategies, we first decided to evaluate the actual sample size that the original S-KNN model would use to sample sessions. We analyzed the average of sampled sessions per recommendation for each dataset, and the results are shown in Figure 7.

Figure 7 – Number of average sampled sessions per recommendations vs maximum sample size. On the left, the chart for the Retailrocket dataset. On the right, the chart values for the dataset NowPlaying.

For the Retailrocket dataset, the average sample size stays below 200 sessions even when the maximum sample size is 1000. This low amount of possible neighbors is probably due to the low average of interactions per session, which limits the possibilities of sessions that interacted with the same items. Although the concentration of interactions by the top 20% most interacted items were very similar between Retailrocket and Nowplaying, the curve of the average sample sizes followed a logarithmic scale, with its peak lower than 343 sessions for maximum sample sizes at 1500. Both results suggest that the sampling strategies are not applied in the majority of cases, which makes the first filtering strategy, which is creating a set of sessions that interacted with the same items of the given session, very efficient to reduce the search space for the S-KNN model. This also means that the second filtering strategy will only be applied either when the item accessed is very popular, hence there would be a large amount of sessions that interacted with the given item, or the session is long, which could aggregate different sets of sessions.

The results presented in (DOMINGOS *et al.*, 2019) also showed that results improved as the number of partitions lowered. This led us to also add the contexts *weekend*, *even/odd hours* and *day/night* to the analysis. With these contexts, we investigated whether the amount of partitions made to the train set interfered in the results. With the context *even/odd hours*, we investigated how a more balanced partition would perform comparing to meaningful partitions such as the *workday/weekend* and *day/night*, given that all contexts divided the dataset in two partitions.

For the initial set of experiments, we set the k -value for the nearest neighbors to 500 and the maximum sample size for the S-KNN model m to 1000 for both datasets, the same parameters used in (LUDEWIG; JANNACH, 2018). However, for the RetailRocket dataset, the average of the relevant sessions for each recommendations was 157.82, which indicated that in most cases the algorithm did not use a sampling strategy to find the nearest neighbors. Therefore, in this set of experiments we fixed $k = 50$ and $m = 100$ for Retailrocket and Nowplaying in order to get a better perspective of the influence of the context as a sampling strategy, but maintained $k = 500$ and $m = 1000$ for RSC15. The average of sampling sessions for each context is shown in Table 7, which shows that the contexts Hour and Weekday greatly reduce the amount of sessions to be considered in the recommendation process, making them much lower compared to k .

Table 7 – Average of sample sessions for each dataset, with $k=50$ and $m = 100$ for Retailrocket and Nowplaying, and $k = 500$ and $m = 1000$ for RSC15.

Context	Retailrocket	Nowplaying	RSC15
Original	58.2	77.0	817.4
Hour	9.3	19.1	264.3
Weekday	22.3	38.6	512.8
Period	37.2	55.6	660.6
Workday/weekend	52.8	71.7	769.0
even/odd hours	43.3	65.6	741.5
day/night	46.3	65.5	687.0
EPCSR	51.9	61.1	746.3

Although this set of experiments used a value of m in Retailrocket ten times lower comparing to the value used in (DOMINGOS *et al.*, 2019), the average amount of samples was still much lower than the current m used, which indicates that, for most interactions, the relevant set of sessions is very small, which severely compromises the capacity of the model to find items of interests. This can also be explained by the average of interactions per item. In Nowplaying, higher average of sampling sessions in all contexts can be explained by the higher average of items per session.

5.2.1.1 Efficacy of temporal contexts as pre-filtering strategies

In this set of experiments, we investigated how the different contexts proposed as pre-filtering strategies affect the performance of the S-KNN algorithm. Simply dividing the datasets into hours of the days and weekdays of the week leads to easy-to-implement pre-filtering strategies, as many programming languages have libraries that retrieve and manipulate dates. On the other hand, these simple strategies significantly deteriorated the performance of the original implementation. However, as the number of partitions shrinks for a given context, the performance of the algorithms improves, as seen in Tables 8, 9 and 10. When applying these set of contexts into the S-KNN extensions S-SKNN, SF-SKNN, and V-SKNN, the performance of the contexts followed the same pattern mentioned previously.

Table 8 – Results of different contexts mentioned in section 4.3 applied to S-KNN as a sampling strategy for the dataset RetailRocket. In this dataset, dividing the dataset in two contexts yielded the best results, with the division by weekdays and weekends having the best performance, followed by dividing the interactions by morning/evening and even/odd hours. The divisions by period of the day, weekdays, and hours had worse performances, as they gradually sliced the dataset into more parts.

Algorithm	HR@20	MRR@20	Cov@20
sknn-workday/weekend	0.5657	0.3353	0.7244
sknn-day/night	0.5478	0.3246	0.6970
sknn-even/odd hours	0.5391	0.3210	0.6834
sknn-period	0.5103	0.3050	0.6433
sknn-weekday	0.4237	0.2596	0.4911
sknn-hour	0.3122	0.1953	0.3090

Table 9 – Results of different contexts mentioned in section 4.3 applied to S-KNN as a sampling strategy for the dataset Nowplaying. In this dataset, the performance of each proposed context followed the same pattern as Table 8, with the exception that in this table, the context even/odd hours performed slightly better than the morning/night context.

Algorithm	HR@20	MRR@20	Cov@20
sknn-workday/weekend	0.2198	0.0598	0.1864
sknn-even/odd hours	0.2104	0.0585	0.1807
sknn-day/night	0.2099	0.0579	0.1791
sknn-period	0.1875	0.0541	0.1693
sknn-weekday	0.1705	0.0491	0.1485
sknn-hour	0.1156	0.0379	0.1175

Table 10 – Results of different contexts mentioned in section 4.3 applied to S-KNN as a sampling strategy for the dataset Nowplaying. In this dataset, the performance of each proposed context followed the same pattern as Table 9.

Algorithm	HR@20	MRR@20	Cov@20
sknn-workday/weekend	0.6237	0.2729	0.6871
sknn-even/odd hours	0.6203	0.2718	0.6708
sknn-day/night	0.6137	0.2697	0.6484
sknn-period	0.6119	0.2691	0.6410
sknn-weekday	0.5757	0.2585	0.5937
sknn-hour	0.5550	0.2507	0.4657

In Tables 11, 12, and 13 we compare the results of each context alongside their original implementations, as well as the EPCSR strategy proposed in (GUO *et al.*, 2019) in the datasets Nowplaying, Retailrocket and RSC15, respectively. The EPCSR strategy performed better in almost all metrics in all datasets, besides having a lower average of sample sessions than the original approach (see Table 7). This reinforces the authors' point that guaranteeing a set of relevant sessions for each item when sampling sessions, especially for the last item, can lead to better performance of neighborhood-based models in session-based recommendations. On Table 13, the original V-SKNN extension had better MRR@20 and Coverage@20 results than

when using EPCSR sampling strategy in RSC15, however, these results were not statistically significant using the two-tailed Wilcoxon Signed-Rank Test with $p < 0.05$.

Table 11 – Results of different contexts mentioned in section 4.3 applied to S-KNN as a sampling strategy for the dataset NowPlaying.

Algorithm	HR@20	MRR@20	Cov@20
sknn-epcsr	0.2433	0.0624	0.2092
sknn	0.2411	0.0637	0.1932
sknn-workday/weekend	0.2198	0.0598	0.1864
sknn-even/odd hours	0.2104	0.0585	0.1807
sknn-day/night	0.2099	0.0579	0.1791
sknn-period	0.1875	0.0541	0.1693
sknn-weekday	0.1705	0.0491	0.1485
sknn-hour	0.1156	0.0379	0.1175
ssknn-epcsr	0.25276	0.06761	0.31745
ssknn	0.24855	0.06757	0.29440
ssknn-workday/weekend	0.22651	0.06356	0.28855
ssknn-day/night	0.21683	0.06242	0.27516
ssknn-even/odd hours	0.21663	0.06225	0.28114
ssknn-period	0.19541	0.05636	0.26109
ssknn-weekday	0.16957	0.05059	0.21918
ssknn-hour	0.11388	0.03874	0.15993
sfsknn-epcsr	0.1727	0.0860	0.2531
sfsknn	0.1701	0.0855	0.2254
sfsknn-workday/weekend	0.1578	0.0753	0.2110
sfsknn-day/night	0.1577	0.0755	0.1979
sfsknn-even/odd hours	0.1556	0.0743	0.1990
sfsknn-period	0.1439	0.0685	0.1744
sfsknn-weekday	0.1300	0.0596	0.1418
sfsknn-hour	0.0931	0.0425	0.0900
vsknn-epcsr	0.2385	0.0639	0.2468
vsknn	0.2340	0.0651	0.2341
vsknn-workday/weekend	0.2132	0.0605	0.2322
vsknn-day/night	0.2006	0.0591	0.2224
vsknn-even/odd hours	0.1986	0.0585	0.2240
vsknn-period	0.1802	0.0536	0.2164
vsknn-weekday	0.1604	0.0490	0.1870
vsknn-hour	0.1058	0.0372	0.1471

5.2.1.2 Discussion

We investigated the performance of the S-KNN model according to the amount of sessions available when finding the nearest neighbors. Having a small but relevant search space is essential for anonymous session-based recommendations, as the system has to yield relevant recommendations during the session without relying on historical information about the user.

Table 12 – Results of different contexts mentioned in section 4.3 applied to S-KNN as a sampling strategy for the dataset Retailrocket.

Algorithm	HR@20	MRR@20	Coverage@20
sknn-epcsr	0.5881	0.3458	0.7608
sknn	0.5793	0.3412	0.7502
sknn-workday/weekend	0.5657	0.3353	0.7244
sknn-day/night	0.5478	0.3246	0.6970
sknn-even/odd hours	0.5391	0.3210	0.6834
sknn-period	0.5103	0.3050	0.6433
sknn-weekday	0.4237	0.2596	0.4911
sknn-hour	0.3122	0.1953	0.3090
ssknn-epcsr	0.5932	0.3493	0.7862
ssknn	0.5833	0.3441	0.7689
ssknn-workday/weekend	0.5694	0.3382	0.7450
ssknn-day/night	0.5522	0.3262	0.7194
ssknn-even/odd hours	0.5408	0.3220	0.7047
ssknn-period	0.5132	0.3066	0.6638
ssknn-weekday	0.4252	0.2612	0.5040
ssknn-hour	0.3128	0.1961	0.3142
sfsknn-epcsr	0.4475	0.2999	0.6626
sfsknn	0.4406	0.2957	0.6442
sfsknn-workday/weekend	0.4340	0.2922	0.6181
sfsknn-day/night	0.4260	0.2857	0.5883
sfsknn-even/odd hours	0.4198	0.2833	0.5730
sfsknn-period	0.4037	0.2725	0.5324
sfsknn-weekday	0.3451	0.2382	0.3975
sfsknn-hour	0.2630	0.1852	0.2511
vsknn-epcsr	0.5767	0.3366	0.7337
vsknn	0.5715	0.3350	0.7220
vsknn-workday/weekend	0.5532	0.3274	0.6994
vsknn-day/night	0.5327	0.3159	0.6776
vsknn-even/odd hours	0.5254	0.3135	0.6632
vsknn-period	0.4930	0.2979	0.6227
vsknn-weekday	0.4101	0.2529	0.4814
vsknn-hour	0.2995	0.1897	0.3073

Surveys such as (WANG; CAO; WANG, 2019; LUDEWIG; JANNACH, 2018) have suggested that incorporating contextual information into the models should be considered as future works for SBRS, and the timestamp of the interactions is a key feature in this scenario, as this feature is used to determine the length of the session and the order of the interactions. Even so, we could also extract information such as period of the day, or day of the week of the sessions from the timestamps, and this information could be used not only by RS, but also for marketing purposes. In this set of experiments, we argued that:

- A sampling strategy will only be needed when the items accessed are very popular or

Table 13 – Average results of different contexts mentioned in section 4.3 applied to S-KNN as a sampling strategy for the dataset RSC 15.

Algorithm	HR@20	MRR@20	Coverage@20
sknn-epcsr	0.6445	0.2792	0.7045
sknn	0.6289	0.2744	0.7021
sknn-workday/weekend	0.6237	0.2729	0.6871
sknn-even/odd hours	0.6203	0.2718	0.6708
sknn-day/night	0.6137	0.2697	0.6484
sknn-period	0.6119	0.2691	0.6410
sknn-weekday	0.5757	0.2585	0.5937
sknn-hour	0.5550	0.2507	0.4657
ssknn-epcsr	0.6661	0.2884	0.7223
ssknn	0.6505	0.2831	0.7195
ssknn-workday/weekend	0.6450	0.2817	0.7053
ssknn-even/odd hours	0.6419	0.2804	0.6886
ssknn-day/night	0.6351	0.2783	0.6660
ssknn-period	0.6334	0.2777	0.6580
ssknn-weekday	0.5953	0.2667	0.6077
ssknn-hour	0.5750	0.2596	0.4725
sfsknn-epcsr	0.6533*	0.2879*	0.7212*
sfsknn	0.6418	0.2831	0.7067
sfsknn-workday/weekend	0.6373	0.2817	0.6871
sfsknn-even/odd hours	0.6358	0.2807	0.6684
sfsknn-day/night	0.6314	0.2789	0.6449
sfsknn-period	0.6305	0.2784	0.6354
sfsknn-weekday	0.5959	0.2680	0.5783
sfsknn-hour	0.5823	0.2614	0.4546
vsknn-epcsr	0.6536	0.2843	0.6978
vsknn	0.6521	0.2853	0.7060
vsknn-workday/weekend	0.6380	0.2799	0.6798
vsknn-even/odd hours	0.6343	0.2785	0.6648
vsknn-day/night	0.6267	0.2764	0.6443
vsknn-period	0.6242	0.2757	0.6364
vsknn-weekday	0.5858	0.2642	0.5914
vsknn-hour	0.5618	0.2552	0.4656

when the sessions are long, since the average of sampled sessions does not scale in the same proportion as the maximum sample size delimited by the user.

- Dividing the dataset into meaningful observational contexts can positively influence the performance of the model. The division of the dataset into workdays and weekends yielded better performances in the contexts proposed. However, when comparing into the other contexts that divided the dataset into 2 contexts, we can infer that this result is caused by the imbalance in the amount of the interactions between contexts.

- Applying a simple observational context filter method can jeopardize the performance of the model, since the separation of the dataset into contexts leaves the recommender with too few examples to yield accurate recommendations. However, a more elaborated sampling strategy such as the EPCSR can yield better recommendations than the original model while using a smaller sample size.

5.2.2 Context in score functions

This set of experiments evaluated the performance of the post-filtering strategies described in Section 4.2.3 for each of the contexts proposed in Section 4.3. Since the amount of possible neighbors per context was very low across the context proposed, we expected that using a post-filtering approach to raise the importance of neighbor sessions that occurred in the same context could yield improvements in the performance of the algorithms. We compared the results of each context for each of the algorithms proposed against the original S-KNN model, S-SKNN and SF-SKNN extensions, as they also change the scoring function of the algorithm. Tables 14, 15 and 16 presents our findings across the datasets used. The V-SKNN model is also considered here as a way to compare an extension in the similarity function against extensions in the scoring method.

The results showed that, although none of the combinations between contexts and scoring methods proposed could outperform the S-KNN and S-SKNN models, in the smaller datasets, the ICF-SKNN model, which elevates the importance of items of sessions that occurred in the same context, could outperform the other extensions used as baselines. In Nowplaying, the contexts with higher average of sampled sessions such as weekday/weekend and morning/night traded accuracy metrics for Coverage gains, which indicates their list of recommendations became slightly more diverse. On the other hand, in the Nowplaying dataset the SF-SKNN had the highest MRR@20 while having low HR@20, which suggests that, for this dataset, the order the songs were played influenced the flow of the session.

On the other hand, the CF-SKNN model, which filters out the influence of the sessions that did not happen in the same context, performed poorly across datasets. However, most contexts could also outperform the SF-SKNN model in the smaller datasets.

5.3 Pre-filter vs Post-Filtering approaches

In this section, we compare the results among the pre-filtering and post-filtering strategies proposed across datasets, shown in Tables 17, 18 and 19. Overall, the ICF-SKNN had the best performances across datasets, and the pre-filtering contexts outperformed the CF-SKNN when comparing the same context. Also, for the ICF-SKNN model, the order of performance for the contexts were inversely proportional to the average number of sampled sessions, whereas the contexts were directly proportional in the other approaches. In other words, the contexts with

Table 14 – Results of different contexts applied to S-KNN as a scoring method for the dataset Nowplaying.

Algorithm	HR@20	MRR@20	Coverage@20
ssknn	0.2486	0.0676	0.2944
sknn	0.2411	0.0637	0.1932
icfsknn-hour	0.2390	0.0633	0.1946
icfsknn-weekday	0.2349	0.0632	0.1988
vsknn	0.2340	0.0651	0.2341
icfsknn-period	0.2255	0.0624	0.2015
icfsknn-even/odd hours	0.2230	0.0613	0.2085
cfsknn-workday/weekend	0.2187	0.0602	0.1970
icfsknn-day/night	0.2141	0.0596	0.2080
cfsknn-day/night	0.2070	0.0581	0.2011
cfsknn-even/odd hours	0.2063	0.0584	0.2008
icfsknn-workday/weekend	0.2058	0.0584	0.2133
cfsknn-period	0.1779	0.0520	0.1943
sfsknn	0.1701	0.0855	0.2254
cfsknn-weekday	0.1427	0.0451	0.1687
cfsknn-hour	0.0822	0.0294	0.1114

Table 15 – Results of different contexts applied to S-KNN as a scoring method for the dataset Retailrocket.

Algorithm	HR@20	MRR@20	Coverage@20
ssknn	0.5833	0.3441	0.7689
sknn	0.5793	0.3412	0.7502
icfsknn-hour	0.5781	0.3410	0.7497
icfsknn-weekday	0.5735	0.3399	0.7471
vsknn	0.5715	0.3350	0.7220
icfsknn-period	0.5650	0.3371	0.7458
cfsknn-workday/weekend	0.5595	0.3343	0.7254
icfsknn-even/odd hours	0.5570	0.3352	0.7423
icfsknn-day/night	0.5546	0.3330	0.7399
icfsknn-workday/weekend	0.5533	0.3285	0.7402
cfsknn-day/night	0.5318	0.3220	0.6966
cfsknn-even/odd hours	0.5195	0.3175	0.6823
cfsknn-period	0.4859	0.3006	0.6369
sfsknn	0.4406	0.2957	0.6442
cfsknn-weekday	0.3765	0.2514	0.4689
cfsknn-hour	0.2435	0.1737	0.2789

the least amount of neighbors in average had the best results in the ICF-SKNN models, but the worse results in the other models proposed.

In RSC15, however, the filtering strategies using the context workday/weekend were among the top performing configurations. For the ICF-SKNN model, this context had the highest Coverage@20.

Table 16 – Average results of different contexts applied to S-KNN as a scoring method for the dataset RSC15.

Algorithm	HR@20	MRR@20	Coverage@20
vsknn	0.6521	0.2853	0.7060
ssknn	0.6505	0.2831	0.7195
sfsknn	0.6418	0.2831	0.7067
sknn	0.6289	0.2744	0.7021
icfknn-hour	0.6232	0.2689	0.7125
cfknn-workday/weekend	0.6213	0.2684	0.6974
icfknn-weekday	0.6207	0.2685	0.7114
icfknn-period	0.6184	0.2681	0.7129
icfknn-day/night	0.6182	0.2680	0.7132
icfknn-even/odd hours	0.6158	0.2677	0.7131
cfknn-even/odd hours	0.6134	0.2667	0.6816
cfknn-day/night	0.6043	0.2648	0.6624
icfknn-workday/weekend	0.6032	0.2646	0.7136
cfknn-period	0.6005	0.2641	0.6560
cfknn-hour	0.4960	0.2372	0.4769
cfknn-weekday	0.3581	0.1706	0.5943

5.4 Final Considerations

This chapter presented the results of the two sets of experiments executed for this work. Our extensive analysis of the datasets and the performance of the S-KNN model allowed us to understand the situations how the sampling strategies could be applied and how they affected the performance of the models.

The conclusions to our hypotheses described in Section ?? are:

1. Based on our experiments, the use of additional contextual features generated from the timestamps of the ongoing session did not provided better recommendations when using neighborhood-based models.
2. Although the extension using temporal contexts as a pre-filtering strategy reduced the search space of neighborhood-based models and extensions, this strategy degraded the performance of the original algorithm.
3. While the extension using temporal contexts as a post-filtering strategy outperformed some sequence-aware extensions proposed by [LUDEWIG; JANNACH](#), this approach could not yield better recommendations than the original S-KNN.

The first set of experiments suggested that having sessions related to all items of the ongoing session is more important than simply filtering out sessions by the context they appeared. In our implementation, we filtered out the sessions in the same context by applying an intersection

Table 17 – Results of different contexts comparing pre-filtering and post-filtering strategies for Retail-Rocket.

Algorithm	HR@20	MRR@20	Coverage@20
icfsknn-hour	0.5781	0.341	0.7497
icfsknn-weekday	0.5735	0.3399	0.7471
sknn-workday/weekend	0.5657	0.3353	0.7244
icfsknn-period	0.565	0.3371	0.7458
cfsknn-workday/weekend	0.5595	0.3343	0.7254
icfsknn-even/odd hours	0.557	0.3352	0.7423
icfsknn-day/night	0.5546	0.333	0.7399
icfsknn-workday/weekend	0.5533	0.3285	0.7402
sknn-day/night	0.5478	0.3246	0.697
sknn-even/odd hours	0.5391	0.321	0.6834
cfsknn-day/night	0.5318	0.322	0.6966
cfsknn-even/odd hours	0.5195	0.3175	0.6823
sknn-period	0.5103	0.305	0.6433
cfsknn-period	0.4859	0.3006	0.6369
sknn-weekday	0.4237	0.2596	0.4911
cfsknn-weekday	0.3765	0.2514	0.4689
sknn-hour	0.3122	0.1953	0.309
cfsknn-hour	0.2435	0.1737	0.2789

between the set of relevant sessions generated in the first step of the S-KNN model and the set of the sessions that happened in the given context. The asymptotic complexity of this operation is linear, where the limit is given by the smaller of the two sets. On the other hand, most sorting algorithms has an average complexity of $n \log n$, where n is the size of the relevant set of sessions. In an application where one of the contexts improved the performance of the algorithm, this extension would be able to generate better recommendations while having reduced computational costs.

Moreover, the second set of experiments showed that the scoring methods proposed combined with contexts with low interactions outperformed some sequence-aware extensions proposed by [LUDEWIG; JANNACH](#). The results also indicate that using contexts to raise the importance of co-occurrence of sessions in a given context is a better strategy than filtering. However, our experiments also suggest that meaningful contexts play an important role, when comparing with meaningless contexts, showing a different result than reported in ([BALTRUNAS; AMATRIAIN, 2009](#)).

Table 18 – Results of different contexts comparing pre-filtering and post-filtering strategies for NowPlaying.

Algorithm	HR@20	MRR@20	Coverage@20
icfsknn-hour	0.239	0.0633	0.1946
icfsknn-weekday	0.2349	0.0632	0.1988
icfsknn-period	0.2255	0.0624	0.2015
icfsknn-even/odd hours	0.223	0.0613	0.2085
sknn-workday/weekend	0.2198	0.0598	0.1864
cfsknn-workday/weekend	0.2187	0.0602	0.197
icfsknn-day/night	0.2141	0.0596	0.208
sknn-even/odd hours	0.2104	0.0585	0.1807
sknn-day/night	0.2099	0.0579	0.1791
cfsknn-day/night	0.207	0.0581	0.2011
cfsknn-even/odd hours	0.2063	0.0584	0.2008
icfsknn-workday/weekend	0.2058	0.0584	0.2133
sknn-period	0.1875	0.0541	0.1693
cfsknn-period	0.1779	0.052	0.1943
sknn-weekday	0.1705	0.0491	0.1485
cfsknn-weekday	0.1427	0.0451	0.1687
sknn-hour	0.1156	0.0379	0.1175
cfsknn-hour	0.0822	0.0294	0.1114

Table 19 – Average results of different contexts comparing pre-filtering and post-filtering strategies for RSC15.

Algorithm	HR@20	MRR@20	Coverage@20
sknn-workday/weekend	0.6237	0.2729	0.6871
icfknn-hour	0.6232	0.2689	0.7125
cfknn-workday/weekend	0.6213	0.2684	0.6974
icfknn-weekday	0.6207	0.2685	0.7114
sknn-even/odd hours	0.6203	0.2718	0.6708
icfknn-period	0.6184	0.2681	0.7129
icfknn-day/night	0.6182	0.268	0.7132
icfknn-even/odd hours	0.6158	0.2677	0.7131
sknn-day/night	0.6137	0.2697	0.6484
cfknn-even/odd hours	0.6134	0.2667	0.6816
sknn-period	0.6119	0.2691	0.641
cfknn-day/night	0.6043	0.2648	0.6624
icfknn-workday/weekend	0.6032	0.2646	0.7136
cfknn-period	0.6005	0.2641	0.656
sknn-weekday	0.5757	0.2585	0.5937
sknn-hour	0.555	0.2507	0.4657
cfknn-hour	0.496	0.2372	0.4769
cfknn-weekday	0.3581	0.1706	0.5943

CONCLUSIONS AND FUTURE WORKS

In the previous chapter we presented experiments made throughout this work, discussing the produced results. In this chapter, we summarize this work, showing the contributions and directions for future works.

6.1 Summary

This work investigated how temporal contexts extracted from the timestamps of the interactions impact the performance of neighborhood-based models in Session-Based Recommender Systems that suggest the next-item of interest to anonymous sessions.

We proposed two different extensions of the model to incorporate the information of time-related contexts in the original S-KNN model. The first model created another step that filters the sessions that co-occurred in the same context of the ongoing session, working as a sampling strategy to reduce the search space in the process of finding the k -nearest neighbors. The second model proposed changes in the scoring method to either filter sessions in different contexts, or raise the importance of sessions that co-occurred in a context.

We presented an extensive analysis of pre-filtering techniques for S-KNN models showing how each pre-filtering strategy affected the average sample of sessions, and how it affects their performances. That allowed us to have a better understanding of how to explore k and m values for future works in S-KNN extensions. We also tested applications of temporal contexts in extensions to the scoring method, both with a post-filtering and a contextual modeling that leverage sessions in the same context, and analyzed how each context influenced the algorithms. Although partitioning past sessions into the temporal contexts did not bring accuracy gains in general, analyzing the datasets through the proposed contexts can offer developers different insights about the usage of their platforms, which can be used to implement different marketing strategies, for example. Besides, this method of recommendation also has an additional layer of

explanation, which can improve the experience of the recommendations.

6.2 Contributions

The main contributions of this work are:

- Two architectures that extends the S-KNN model by using timestamps of the interactions to group sessions into different contexts that can be used in pre-filtering and post-filtering strategies, available in <<https://github.com/mrtakata/session-recommender>>;
- An extensive statistical analysis of standard datasets used for researches in SBRS;
- An strategy to optimize the k -value and the sample size for S-KNN models;
- Evaluation of the impact of extracted temporal features from timestamps both in pre-processing and scoring methods of S-KNN models;

Furthermore, during the development of this work, we were able to publish a work-in-progress (DOMINGOS *et al.*, 2019):

Title: Exploring temporal contexts for neighborhood-based models in Session-Based Recommender Systems

Authors: Gabriel B. Domingos, Igor Lovatto, Matheus Takata and Marcelo G. Manzato

Conference: Workshop de trabalhos de iniciação científica - Simpósio Brasileiro de Sistemas Multimídia e Web (Webmedia)

Year: 2019

Local: Rio de Janeiro, RJ, Brazil

Abstract: Recent works in the field of Recommender Systems have focused in making recommendations using only information about accessed items in an active user session, since the user's preferences can be very specific to the context of the session. Despite the increasing interest in deep neural network models in the Recommender Systems in general, simpler models, such as neighborhood-based collaborative filtering ones, have been outperforming these more complex models in the session-based scenario. However, in order to soften known scalability problems in neighborhood-based models, sampling strategies during the recommendation process have been proposed, showing that some session contexts such as recency of neighbor sessions can improve the recommendation in some domains. Therefore this project explores the effect of different temporal contexts that can easily be captured from an anonymous user session, such as the hour of the day and day of the week, as alternatives to the established sampling methods.

6.3 Limitations

With advancements of Artificial Intelligence and wide usage of personal data by the platforms, privacy concerns have to be considered when building a Recommender System for a platform. This work was developed using only the IDs of anonymous sessions and the items, as well as the timestamps of the interactions among them. This strategy allows these methods to be easily applicable in scenarios where the user is not logged in the platform, either because they are a newcomer or they do not want to be identified in a platform. However, not having human-understandable data makes the process of understanding the data, and therefore, planning new recommendation strategies more difficult. In fact, some datasets used for research in SBRS systems even mask the timestamps, making it only possible to infer the order of interactions, and the time a user spends with an item. Therefore, these characteristics must be kept in mind when developing academic research in this area.

Since we wanted to explore a variety of time-related contexts, offline experiments allowed us to easily propose and evaluate different contexts to analyze which cases contexts could yield better results in a real-life scenario. However, our analysis were limited by the domains of the datasets and the characteristics of the systems. An online experiment could have more agency with dealing with factors outside the recommendation process, besides the ability to further investigate the quality of the recommendations with the users and content providers.

6.4 Future Works

The conclusion of this work indicates that there is still room for improvements and exploration for applying context into neighborhood models. Therefore, this section suggests future works that can be developed after this work:

- Use more nuanced contexts from timestamps, for example, holiday seasons, divide periods of day in working/non-working hours;
- build hybrid models that combines different time-related contexts;
- Aggregate different types of feedback into the model, such as buying and add-to-cart interactions when available;
- Use decay functions in scoring functions based on the extracted contexts;
- Develop new scoring methods that raise the importance of co-occurrence of sessions in a context;
- Evaluate the performance of the TARS approaches using the designed contexts in different models;

- Evaluate the algorithms in an online environment, so other features such as novelty of an item, or the organization of the recommended items on a platform can also be analyzed.

BIBLIOGRAPHY

ADOMAVICIUS, G.; TUZHILIN, A. Context-aware recommender systems. In: _____. **Recommender Systems Handbook**. Boston, MA: Springer US, 2011. p. 217–253. ISBN 978-0-387-85820-3. Available: <https://doi.org/10.1007/978-0-387-85820-3_7>. Citations on pages 22 and 32.

AGGARWAL, C. C. Content-based recommender systems. In: _____. **Recommender Systems: The Textbook**. Cham: Springer International Publishing, 2016. p. 139–166. ISBN 978-3-319-29659-3. Available: <https://doi.org/10.1007/978-3-319-29659-3_4>. Citation on page 30.

_____. Ensemble-based and hybrid recommender systems. In: _____. **Recommender Systems: The Textbook**. Cham: Springer International Publishing, 2016. p. 199–224. ISBN 978-3-319-29659-3. Available: <https://doi.org/10.1007/978-3-319-29659-3_6>. Citation on page 31.

_____. An introduction to recommender systems. In: _____. **Recommender Systems: The Textbook**. Cham: Springer International Publishing, 2016. p. 1–28. ISBN 978-3-319-29659-3. Available: <https://doi.org/10.1007/978-3-319-29659-3_1>. Citation on page 25.

_____. Knowledge-based recommender systems. In: _____. **Recommender Systems: The Textbook**. Cham: Springer International Publishing, 2016. p. 167–197. ISBN 978-3-319-29659-3. Available: <https://doi.org/10.1007/978-3-319-29659-3_5>. Citation on page 30.

_____. Neighborhood-based collaborative filtering. In: _____. **Recommender Systems: The Textbook**. Cham: Springer International Publishing, 2016. p. 29–70. ISBN 978-3-319-29659-3. Available: <https://doi.org/10.1007/978-3-319-29659-3_2>. Citations on pages 26 and 27.

_____. Time- and location-sensitive recommender systems. In: _____. **Recommender Systems: The Textbook**. Cham: Springer International Publishing, 2016. p. 283–308. ISBN 978-3-319-29659-3. Available: <https://doi.org/10.1007/978-3-319-29659-3_9>. Citations on pages 33 and 34.

BALTRUNAS, L.; AMATRIAIN, X. Towards time-dependant recommendation based on implicit feedback. p. 25–30, 2009. Citations on pages 32, 33, 52, 54, 58, and 74.

Barkan, O.; Koenigstein, N. Item2vec: Neural item embedding for collaborative filtering. p. 1–6, 2016. Citations on pages 22, 27, and 43.

BOGINA, V.; KUFLIK, T. Incorporating dwell time in session-based recommendations with recurrent neural networks. p. 57–59, 2017. Citations on pages 50 and 52.

BOOM, C. D.; AGRAWAL, R.; HANSEN, S.; KUMAR, E.; YON, R.; CHEN, C.-W.; DE-MEESTER, T.; DHOEDT, B. Large-scale user modeling with recurrent neural networks for music discovery on multiple time scales. **Multimedia Tools and Applications**, Springer, v. 77, n. 12, p. 15385–15407, 2018. Citation on page 21.

BRYNJOLFSSON, E.; HU, Y.; SIMESTER, D. Goodbye pareto principle, hello long tail: The effect of search costs on the concentration of product sales. **Management Science**, INFORMS, v. 57, n. 8, p. 1373–1386, 2011. Citation on page 62.

COSTA, A. F. da; MANZATO, M. G.; CAMPELLO, R. J. Group-based collaborative filtering supported by multiple users' feedback to improve personalized ranking. In: **Proceedings of the 22nd Brazilian Symposium on Multimedia and the Web**. New York, NY, USA: Association for Computing Machinery, 2016. (Webmedia '16), p. 279–286. ISBN 9781450345125. Available: <https://doi.org/10.1145/2976796.2976852>. Citations on pages 27 and 37.

DEY, A. K. Understanding and using context. **Personal Ubiquitous Comput.**, Springer-Verlag, London, UK, UK, v. 5, n. 1, p. 4–7, Jan. 2001. ISSN 1617-4909. Available: <http://dx.doi.org/10.1007/s007790170019>. Citation on page 31.

DOMINGOS, G. B.; LOVATTO, I.; TAKATA, M.; MANZATO, M. Exploring temporal contexts for neighborhood-based models in session-based recommender systems. In: **Anais Estendidos do XXV Simpósio Brasileiro de Sistemas Multimídia e Web**. Porto Alegre, RS, Brasil: SBC, 2019. p. 55–58. ISSN 2596-1683. Available: https://sol.sbc.org.br/index.php/webmedia_estendido/article/view/8136. Citations on pages 33, 64, 65, 66, and 78.

DOURISH, P. What we talk about when we talk about context. **Personal and ubiquitous computing**, Springer, v. 8, n. 1, p. 19–30, 2004. Citation on page 32.

D'ADDIO, R. M.; DOMINGUES, M. A.; MANZATO, M. G. Exploiting feature extraction techniques on users' reviews for movies recommendation. **Journal of the Brazilian Computer Society**, v. 23, n. 1, p. 7, 2017. Citations on pages 29, 31, and 37.

Forsati, R.; Meybodi, M. R.; Neiat, A. G. Web page personalization based on weighted association rules. p. 130–135, 2009. Citation on page 39.

GARG, D.; GUPTA, P.; MALHOTRA, P.; VIG, L.; SHROFF, G. Sequence and time aware neighborhood for session-based recommendations: Stan. In: **Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval**. New York, NY, USA: Association for Computing Machinery, 2019. (SIGIR'19), p. 1069–1072. ISBN 9781450361729. Available: <https://doi.org/10.1145/3331184.3331322>. Citations on pages 33, 37, 49, 51, and 52.

GEMMIS, M. de; LOPS, P.; MUSTO, C.; NARDUCCI, F.; SEMERARO, G. Semantics-aware content-based recommender systems. In: _____. **Recommender Systems Handbook**. Boston, MA: Springer, 2015. p. 119–159. Citations on pages 29 and 30.

GUNAWARDANA, A.; SHANI, G. Evaluating recommender systems. In: _____. **Recommender Systems Handbook**. Boston, MA: Springer, 2015. p. 265–308. Citation on page 35.

GUO, H.; TANG, R.; YE, Y.; LIU, F.; ZHANG, Y. A novel knn approach for session-based recommendation. In: YANG, Q.; ZHOU, Z.-H.; GONG, Z.; ZHANG, M.-L.; HUANG, S.-J. (Ed.). **Advances in Knowledge Discovery and Data Mining**. Cham: Springer International Publishing, 2019. p. 381–393. ISBN 978-3-030-16145-3. Citations on pages 22, 27, 36, 37, 48, 51, 52, 54, 61, 64, and 67.

HIDASI, B.; KARATZOGLOU, A.; BALTRUNAS, L.; TIKK, D. Session-based recommendations with recurrent neural networks. **CoRR**, abs/1511.06939, 2015. Available: <http://arxiv.org/abs/1511.06939>. Citations on pages 22, 36, and 37.

HIDASI, B.; QUADRANA, M.; KARATZOGLOU, A.; TIKK, D. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In: **Proceedings of the 10th ACM Conference on Recommender Systems**. New York, NY, USA: ACM, 2016. (RecSys

'16), p. 241–248. ISBN 978-1-4503-4035-9. Available: <<http://doi.acm.org/10.1145/2959100.2959167>>. Citations on pages 43, 61, 62, and 64.

JANNACH, D.; LUDEWIG, M. When recurrent neural networks meet the neighborhood for session-based recommendation. In: **Proceedings of the Eleventh ACM Conference on Recommender Systems**. New York, NY, USA: Association for Computing Machinery, 2017. (RecSys '17), p. 306–310. ISBN 9781450346528. Available: <<https://doi.org/10.1145/3109859.3109872>>. Citations on pages 27, 39, 44, 46, and 47.

KAMINSKAS, M.; BRIDGE, D. Diversity, serendipity, novelty, and coverage: A survey and empirical analysis of beyond-accuracy objectives in recommender systems. **ACM Transactions on Interactive Intelligent Systems (TiIS)**, ACM, v. 7, n. 1, p. 2, 2016. Citation on page 36.

KOREN, Y. Collaborative filtering with temporal dynamics. **Communications of the ACM**, ACM, v. 53, n. 4, p. 89–97, 2010. Citations on pages 28 and 33.

_____. Collaborative filtering with temporal dynamics. **Communications of the ACM**, ACM, v. 53, n. 4, p. 89–97, 2010. Citation on page 33.

KOREN, Y.; BELL, R. Advances in collaborative filtering. In: _____. **Recommender systems handbook**. Boston, MA: Springer, 2015. p. 77–118. Citation on page 28.

KOREN, Y.; BELL, R.; VOLINSKY, C. Matrix factorization techniques for recommender systems. **Computer**, IEEE, v. 42, n. 8, 2009. Citations on pages 27 and 28.

LI, J.; REN, P.; CHEN, Z.; REN, Z.; LIAN, T.; MA, J. Neural attentive session-based recommendation. In: **Proceedings of the 2017 ACM on Conference on Information and Knowledge Management**. New York, NY, USA: Association for Computing Machinery, 2017. (CIKM '17), p. 1419–1428. ISBN 9781450349185. Available: <<https://doi.org/10.1145/3132847.3132926>>. Citation on page 43.

LI, X.; JIANG, M.; HONG, H.; LIAO, L. A time-aware personalized point-of-interest recommendation via high-order tensor factorization. **ACM Transactions on Information Systems (TOIS)**, ACM New York, NY, USA, v. 35, n. 4, p. 1–23, 2017. Citation on page 33.

LUDEWIG, M.; JANNACH, D. Evaluation of session-based recommendation algorithms. **User Modeling and User-Adapted Interaction**, v. 28, n. 4, p. 331–390, Dec 2018. ISSN 1573-1391. Available: <<https://doi.org/10.1007/s11257-018-9209-6>>. Citations on pages 22, 23, 27, 32, 33, 36, 37, 40, 42, 43, 47, 50, 51, 52, 53, 54, 61, 62, 64, 66, 69, 73, and 74.

MANZATO, M. G. gsvd++: supporting implicit feedback on recommender systems with meta-data awareness. p. 908–913, 2013. Citations on pages 28 and 31.

MI, F.; FALTINGS, B. **Context Tree for Adaptive Session-based Recommendation**. 2018. Citations on pages 43, 50, and 52.

MIKOLOV, T.; CHEN, K.; CORRADO, G.; DEAN, J. Efficient estimation of word representations in vector space. **arXiv preprint arXiv:1301.3781**, 2013. Citation on page 31.

MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G. S.; DEAN, J. Distributed representations of words and phrases and their compositionality. p. 3111–3119, 2013. Citation on page 30.

MUSTO, C.; SEMERARO, G.; GEMMIS, M. de; LOPS, P. Learning word embeddings from wikipedia for content-based recommender systems. p. 729–734, 2016. Citation on page 30.

NING, X.; DESROSIERS, C.; KARYPIS, G. A comprehensive survey of neighborhood-based recommendation methods. In: _____. **Recommender systems handbook**. Boston, MA: Springer, 2015. p. 37–76. Citation on page 27.

NOIA, T. D.; MIRIZZI, R.; OSTUNI, V. C.; ROMITO, D.; ZANKER, M. Linked open data to support content-based recommender systems. p. 1–8, 2012. Citation on page 30.

PALUMBO, E.; RIZZO, G.; TRONCY, R. Entity2rec: Learning user-item relatedness from knowledge graphs for top-n item recommendation. In: **ACM. Proceedings of the Eleventh ACM Conference on Recommender Systems**. Como, Italy, 2017. p. 32–36. Citation on page 27.

QUADRANA, M.; CREMONESI, P.; JANNACH, D. Sequence-aware recommender systems. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 51, n. 4, p. 1–36, 2018. Citation on page 34.

QUADRANA, M.; KARATZOGLOU, A.; HIDASI, B.; CREMONESI, P. Personalizing session-based recommendations with hierarchical recurrent neural networks. In: **Proceedings of the Eleventh ACM Conference on Recommender Systems**. New York, NY, USA: Association for Computing Machinery, 2017. (RecSys '17), p. 130–137. ISBN 9781450346528. Available: <<https://doi.org/10.1145/3109859.3109896>>. Citation on page 43.

RUOCCO, M.; SKREDE, O. S. L.; LANGSETH, H. Inter-session modeling for session-based recommendation. In: **Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems**. Como, Italy: Association for Computing Machinery, 2017. p. 24–31. Citation on page 51.

SMITH, B.; LINDEN, G. Two decades of recommender systems at amazon. com. **IEEE internet computing**, IEEE, v. 21, n. 3, p. 12–18, 2017. Citation on page 21.

STEFANCOVA, E.; SRBA, I. Poi recommendation based on locality-specific seasonality and long-term trends. In: CHATZIGEORGIOU, A.; DONDI, R.; HERODOTOU, H.; KAPOUTSIS, C.; MANOLOPOULOS, Y.; PAPADOPOULOS, G. A.; SIKORA, F. (Ed.). **SOFSEM 2020: Theory and Practice of Computer Science**. Cham: Springer International Publishing, 2020. p. 338–349. ISBN 978-3-030-38919-2. Citations on pages 50, 52, and 54.

TAN, Y. K.; XU, X.; LIU, Y. Improved recurrent neural networks for session-based recommendations. In: **Proceedings of the 1st Workshop on Deep Learning for Recommender Systems**. Boston, MA: Association for Computing Machinery, 2016. p. 17–22. Citation on page 50.

WANG, S.; CAO, L.; WANG, Y. **A Survey on Session-based Recommender Systems**. 2019. Citations on pages 13, 39, 40, 43, and 69.

ZANGERLE, E.; PICHL, M.; GASSLER, W.; SPECHT, G. # nowplaying music dataset: Extracting listening behavior from twitter. In: **Proceedings of the first international workshop on internet-scale multimedia management**. Orlando Florida USA: ACM, 2014. p. 21–26. Citation on page 62.

ZHANG, C.; CHENG, X. An ensemble method for job recommender systems. In: **Proceedings of the Recommender Systems Challenge**. New York, NY, USA: Association for Computing Machinery, 2016. (RecSys Challenge '16). ISBN 9781450348010. Available: <<https://doi.org/10.1145/2987538.2987545>>. Citation on page 31.

ZHOU, T.; REN, J.; MEDO, M.; ZHANG, Y.-C. Bipartite network projection and personal recommendation. **Physical review E**, APS, v. 76, n. 4, p. 046115, 2007. Citation on page 34.