Pontifícia Universidade Católica do Paraná
Programa de Pós-Graduação em Informática

Latent Vectors' Update Optimization and Negatives-Relevant Sampling in One Class Collaborative Filtering Recommender Systems

Antônio David Viniski

Supervisor

Jean Paul Barddal

Co-Supervisor

Alceu de Souza Britto Jr.

Curitiba
2023

Pontifícia Universidade Católica do Paraná
Programa de Pós-Graduação em Informática

Latent Vectors' Update Optimization and Negatives-Relevant Sampling in One Class Collaborative Filtering Recommender Systems

Antônio David Viniski

Thesis presented to the *Programa de Pós-Graduação em Informática* as a partial requirement for the degree of Doctor in Informatics.

Major Field: Computer Science

Supervisor: Jean Paul Barddal

Co-supervisor: Alceu de Souza Britto Jr.

Curitiba
2023

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Sônia Maria Magalhães da Silva – CRB 9/1191

Viniski, Antônio David
V785l Latent vectors' update optimization and negatives-relevant sampling in one
2023 class collaborative filtering recommender systems / Antônio David Viniski ;
supervisor: Jean Paul Barddal ; co-supervisor: Alceu de Souza Britto Jr. – 2023
xiii, 131 f. ; il. : 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2023
Bibliografia: f. 107-120

1. Fluxo de dados (Computadores). 2. Processamento eletrônico de dados.
3. Informática. I. Barddal, Jean Paul. II. Britto Júnior, Alceu de Souza. III.
Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em
Informática. IV. Título.

CDD. 20. ed. – 004

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

Curitiba, 07 de novembro de 2023.

86-2023

DECLARAÇÃO

Declaro para os devidos fins, que **ANTÔNIO DAVID VINISKI** defendeu a tese intitulada "**Latent Vectors' Update Optimization and Negatives-Relevant Sampling in One Class Collaborative Filtering Recommender Systems"**, na área de concentração Ciência da Computação no dia 30 de junho de 2023, no qual foi aprovado.

Declaro ainda, que foram feitas todas as alterações solicitadas pela Banca Examinadora, cumprindo todas as normas de formatação definidas pelo Programa.

Por ser verdade firmo a presente declaração.

_____
Prof. Dr. Emerson Cabrera Paraiso
Coordenador do Programa de Pós-Graduação em Informática

Dedico este trabalho a todos que contribuíram durante o meu processo de doutoramento: a minha família, por me apoiar e incentivar na busca pelo conhecimento; aos meus mestres, que confiaram em mim, sempre me auxiliaram e continuam auxiliando dentro e fora do meio acadêmico; e aos meus amigos, que vibram comigo em cada conquista.

"Eu prefiro ter perguntas que não podem ser respondidas a ter respostas que não podem ser questionadas". Richard Feynman

Abstract

Recommender systems suggest items that a particular user prefers based on historical behavior, actions, and feedback. Data on users and items are rapidly and continuously generated in real-world applications, such as e-commerce, social media, and digital marketing. Thus, these scenarios can be formulated as data stream problems. Furthermore, the recommendation systems must deal with challenges inherent in data streams, such as concept drifts and cold start, since the users' interests are dynamic and change over time, and the emergence of new users and items from the stream is also expected. Several techniques have been proposed in the literature to tackle these recommendation problems. However, most of these techniques require additional information about users and items, such as demographic and social network data. Nonetheless, additional information is often unavailable in most real-world scenarios, and the system must use the provided data to alleviate these problems. In this sense, this project aims to optimize collaborative recommendation models via adaptations in latent vector updates and unobserved negative-relevant item sampling. As a byproduct of this research, a novel recommendation dataset for collaborative filtering was introduced, in which was tested popular batch and streaming recommendation models. It was proposed four variants of four adaptive learning rate optimizers, Adam, AMSGrad, Nadam, and RMSprop, to provide more significant updates in the Incremental MF recommender models. In addition, one of our contributions relies on optimizing the property unknown items selection in models trained with negative items. It was proposed two negative-relevant sampling strategies combined with pairwise and point-wise recommender models. According to the results obtained, both contributions significantly improve the recommender model's performance. Were observed that the incremental models are most suitable in scenarios with concept drift and cold-start in the introduced dataset. The adaptive optimizers improve the RECALL and NDCG rates by up to 11.1 and 7.5 percentage points, respectively. Considering the negative-relevant sampling contribution, the proposed methods increased RECALL values by up to 17.4 percentage points.
Key-words: Streaming Recommendation, Incremental Learning, Collaborative Filtering, Implicit Feedback

Resumo

Os sistemas de recomendação sugerem itens que um determinado usuário prefere com base em seu comportamento histórico, ações e feedback. Em aplicativos do mundo real, dados sobre usuários e itens são gerados de forma contínua e rápida, como e-commerce, mídia social, marketing digital e consumo de conteúdo em muitos domínios. Portanto, esses cenários podem ser formulados como um problema de fluxo de dados. Além disso, os sistemas de recomendação devem lidar com alguns desafios da moda streaming, como concept drifts e cold start, uma vez que os interesses dos usuários são dinâmicos e mudam com o tempo, e também é esperado o surgimento de novos usuários e itens do fluxo. Várias técnicas têm sido propostas na literatura para resolver esses problemas de recomendação. No entanto, a maioria dessas técnicas requer informações adicionais sobre usuários e itens, como dados demográficos e de redes sociais. No entanto, informações adicionais geralmente não estão disponíveis na maioria dos cenários do mundo real, e o sistema deve usar os dados fornecidos para aliviar esses problemas. Nesse sentido, este projeto visa otimizar modelos de recomendação colaborativa por meio de adaptações em atualizações de vetores latentes e amostragem de itens negativos relevantes não observados. Como subproduto desta pesquisa, um novo conjunto de dados de recomendação para filtragem colaborativa é introduzido, no qual foram testados modelos populares de recomendação em lote e streaming. Foram propostas quatro variantes de quatro otimizadores de taxa de aprendizado adaptativo, Adam, AMSGrad, Nadam e RMSprop, para fornecer atualizações mais significativas nos modelos de recomendação incremental baseados na Fatoração de Matriz. Além disso, uma das contribuições dessa pesquisa se baseia na otimização da seleção apropriada de itens desconhecidos em modelos treinados com itens negativos. Foram propostas duas estratégias de amostragem de itens negativos e relevantes, as quais foram combinadas com modelos de recomendação pairwise e point-wise. De acordo com os resultados obtidos, ambas as contribuições são relevantes para a comunidade de pesquisa e melhoram significativamente o desempenho do modelo de recomendação. Pode-se observar que os modelos incrementais são mais adequados em cenários com concept-drifts e cold-start no conjunto de dados proposto. Os otimizadores adaptativos melhoram as taxas de RECALL e NDCG em até 11,1 e 7,5 pontos percentuais, respectivamente. Considerando a

contribuição de amostragem de itens negativos-relevantes, os métodos propostos aumentaram os valores de RECALL em até 17,4 pontos percentuais. Como o foco da pesquisa são os cenários streaming, em trabalhos futuros, planeja-se incorporar detectores de mudanças em nossas estratégias propostas. Além disso, pretende-se testar outras estratégias de seleção de itens desconhecidos para comparação.
Palavras-chave: Recomendação em fluxo de dados, Aprendizagem Incremental, Filtragem Colaborativa, Feedback Implícito

Acknowledgements

I first thank my family, my parents, Antônio and Rosilda, my sister Luana, and my brother-in-law Alessandro for all the support and dedication during the development of the doctoral project. You were indispensable at this stage of my life and continue to be, so I certainly intend to share many other achievements with you.

I want to thank the other family members, who, even though they were far away, cheered for me, put their prayers on me, and made this moment even more special.

I thank my dear friends, who even indirectly contributed to the success of this stage. In particular, I would like to thank my colleagues from laboratory 26, Bruno, Rodrigo, Bruna, and Marcos, for all the exchange of knowledge that we had during this period.

I thank all PUCPR employees for their support, faculty, secretariat, coordination, security, support, etc.

Finally, I am immensely grateful to my advisors, Jean and Alceu, for making this moment possible, trusting me, embarking on my ideas, and carefully guiding me along the best paths. You were fundamental for the completion of this project.

Agradecimentos

Agradeço primeiramente a minha família, meus pais Antônio e Rosilda, minha irmã Luana, meu cunhado Alessandro, por todo o apoio e dedicação durante o desenvolvimento do projeto de doutorado. Vocês foram indispensáveis nesta etapa da minha vida e continuam sendo, por isso pretendo com certeza compartilhar de muitas outras alegrias com vocês.

Agradeço aos demais familiares, que mesmo estando longe torceram por mim, me colocaram eu suas orações e fizeram desse momento ainda mais especial.

Aos meus queridos amigos, que mesmo indiretamente contribuíram para o sucesso dessa etapa. Em especial, agradeço os meus colegas do laboratório 26, Bruno, Rodrigo, Bruna e Marcos, por toda troca de conhecimentos que tivemos nesse período. Também sou muito grato aos meus amigos Leandro, Rafael, Fábio, David, Alessandro e todos os demais que suportaram comigo os momentos de angústia e estiveram ao meu lado, mesmo que distantes, durante esse percurso.

A todos os funcionários da PUCPR pelo apoio prestado, docentes, secretaria, coordenação, segurança, suporte, etc.

Por fim, agradeço imensamente os meus orientadores, Jean e Alceu, por tornarem esse momento possível, confiaram em mim, embarcaram nas minhas ideias e cuidadosamente me guiaram pelos melhores caminhos, vocês foram fundamentais para a finalização desde projeto.

Contents

List of Figures

List of Tables

List of Algorithms

1

INTRODUCTION

Due to the increasing amount of data collected, recommendation systems have proven to be indispensable in several scenarios where defining user profiles is essential. The modeling of consumption patterns enables personalized and relevant recommendations for users (RICCI; ROKACH; SHAPIRA, 2011). The personalization of actions has become an important marketing tool in numerous areas (ZHANG et al., 2019). Companies such as Amazon, Netflix, and Google News stand out using efficient recommendation methods, seeking to improve the user experience on their systems and enable the engagement of their products and services assertively (LI et al., 2017a).

Most of the developed recommendation models work in a batch fashion. Thus, given a training set consisting of interactions between users and items, a static model is generated and later used in the recommendation process (BABÜROGLU; DURMUSOGLU; DERELI, 2021). Consequently, an important research topic deals with adapting recommendation systems to work incrementally, assuming that interactions between users and items are made available as a data stream of events (VINISKI et al., 2021).

According to the available data and the recommendation process characteristics, the research area classifies the recommendation models traditionally into three categories: Content-Based Filtering, Collaborative Filtering, and Hybrid Filtering (some authors suggest other subdivisions, such as Session-Based Filtering and Knowledge-Based Filtering) (ZHANG et al., 2019). In Collaborative Filtering, the methods use only a list of interactions between users and items to generate the models, while in Content-Based Filtering, details about the

items are necessary. Consequently, Collaborative Filtering is less restrictive and has been the subject of many works over the years (BOBADILLA et al., 2013). In the collaborative recommendation systems, users' feedback, which includes assessments, transactions, interactions, and examinations, is considered critical information to learn and model users' preferences and can be obtained explicitly or implicitly (PIRAMUTHU et al., 2012).

Explicit feedback is usually represented by numeric grades with different levels (Netflix 1-5 scale, for example). However, collecting explicit user feedback is difficult, complicated, and often costly, as it requires new functionality in the systems (PAN; LIU; MING, 2016; VINAGRE; JORGE; GAMA, 2014). On the other hand, the implicit *feedback* expressed by users, such as watching a movie, listening to a song, enjoying a post, buying a product, and so forth, is easy to obtain and has attracted growing interest from researchers (VINAGRE; JORGE; GAMA, 2014; LI et al., 2017a).

Although the advantages are clear, implicit feedback, referred to as One-Class Collaborative Filtering (OCCF) and Positive-only Feedback, presents several challenging characteristics. The data sparsity refers to the number of unobserved interactions and the absence of negative feedback since only positive observations are available. These are two examples of critical problems in the one-class collaborative scenario. In addition to the abovementioned problems, many other challenges related to the collaborative recommendation process are the hot topics in last year's research. This thesis focus on three specific challenges that affect the recommendation model's performance: negative item selection, concept drift, and cold-start.

The first challenge deals with the possibility of using the unobserved interactions (unknown relations in the user-item matrix) as negative ones. Using negative interactions in the training stage improves the recommender models' convergence process and reduces data imbalance (PAN et al., 2008). On the other hand, most of the existing works select negative items randomly. Such selection can choose relevant items as negatives, reducing the model's accuracy (YU; BILENKO; LIN, 2017). The second challenge, concept drift, refers to the changes in data behavior over time (WEBB et al., 2018). In recommendation systems, concept drifts reflect changes between customers and products interactions, whether because (i) user preferences change, (ii) new items are made available, (iii) specific offers at different time granularities (days of the week, weeks of the month, months of the year), and so forth. Finally, the third

challenge, called cold-start, occurs when new users or items appear in the recommendation scenarios (WEI et al., 2017).

Using models capable of continuously modifying its parameters, considering the changes in the data's behavior, is a computational alternative to reduce the impacts on the performance of the recommender models caused by the cold-start and concept drift problems (VINISKI et al., 2021). Considering scenarios in which the data present continuous changes due to the appearance of new users and new items, or even by specific systems' rules (such as promotions, actions in periods of the year, month, or week), the recommendation models must be able to identify these changes and adapt to maintain performance.

1.1 Objectives

This work aims to develop adaptive recommender models by generating new approaches for updating latent factors and selecting negative-relevant items, seeking to reduce the improved recommendations in implicit datasets.

The specific objectives of this project are:

- Collect, process and make available new retail datasets for OCCF with concept drift and cold-start.
- Define a comparison protocol between batch processing and data stream models.
- Analyze and implement existing adaptive and incremental recommendation models.
- Study the approaches to adapting the learning rate in data stream scenarios seeking to improve recommendations.
- Implement adaptive recommendation models by combining machine learning optimizers to adapt the models based on user and item characteristics.
- Implement negative-relevant items sampling alternative to the traditional uniform random sampling of negative items.
- Compare the developed methods with existing models in the literature.

1.2 Hypotheses

This section presents our hypotheses related to the proposed recommendation challenges:

Hypothesis #1. Adaptive and incremental methods present superior performance compared with batch processing techniques in datasets with concept drift and a high incidence of cold-start.

Hypothesis #2. Utilizing learning rate adaptation techniques improves the recommender models' performance in streaming scenarios.

Hypothesis #3. Utilizing methods that consider the dataset characteristics to select negative and relevant items in the recommendation techniques training allows better model adjustment than random negative selection.

1.3 Contributions

In addition to providing new adaptive recommendation approaches, this thesis presents novel datasets and robust experimental protocols for comparing the proposed techniques with traditional techniques in the literature. Thus, the main contributions are the following:

- Provide real-world datasets from retail containing cold-start and concept-drift characteristics to support the development and evaluation of adaptive and incremental recommendation methods.
- Introduce a robust experimental protocol to compare traditional batch and streaming recommender models.
- Develop adaptive recommendation techniques for incremental streaming scenarios.
- Define incremental parameters update methods to improve the efficiency of the recommender models.
- Generate negative-relevant item selection strategies for training recommender systems with implicit datasets.

1.4 Publications

The main results of this thesis are reported in the following publications.

- Antônio David Viniski, Jean Paul Barddal, Alceu de Souza Britto Jr., Fabrício Enembreck, Humberto Vinicius Aparecido de Campos. A case study of batch and incremental recommender systems in supermarket data under concept drifts and cold start. In: Expert Systems with Applications, Volume 176, Pages 114890, 2021. <https://doi.org/10.1016/j.eswa.2021.114890>
- Antônio David Viniski, Jean Paul Barddal, Alceu de Souza Britto Jr. UKIRF: An Item Rejection Framework for Collaborative Filtering. In: Karlapalem K. et al. (eds) Advances in Knowledge Discovery and Data Mining. PAKDD 2021. Lecture Notes in Computer Science, vol 12713. Springer, Cham, 2021. <https://doi.org/10.1007/978-3-030-75765-6_44>
- **(2nd revision - Under Review - Last review activity: 11th April 2023)** Antônio David Viniski, Jean Paul Barddal, Alceu de Souza Britto Jr., Humberto Vinicius Aparecido de Campos. Incremental Specialized and Specialized-Generalized Matrix Factorization Models based on Adaptive Learning Rate Optimizers. In: Neurocomputing.

1.5 Financial Support

This thesis is financially supported by the *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq) under the Grant Number #142195/2019-7. In addition, this project is characterized as an Academic Doctorate for Innovation (*Doutorado acadêmico para inovação - DAI*) and is part of technical cooperation between the *Pontifícia Universidade Católica do Paraná (PUCPR)* and the *Himarket Consultoria De Comunicação E Ti Ltda (Himarket)* company.

1.6 Overview

This thesis is organized into three parts. The first part presents the background of the recommendations research area and the related work.

- Chapter 2 introduces the learning schemes, recommendation approaches, problems, and popular available datasets.
- Chapter 3 presents popular batch and streaming recommendation techniques.

The second part introduces this work's main contributions, surrounding the availability of a novel retail dataset, the recommender model's optimization strategies, and the negative-relevant sampling strategies.

- Chapter 5 introduces a novel dataset collection containing three variants of a supermarket dataset, two preprocessing strategies, and their characteristics and content.
- Chapter 6 shows the proposed optimizers variants to replace the traditional Stochastic Gradient Descent (SGD) optimizer in the incremental MF model.
- Chapter 7 presents two relevant items generation strategies combined with the pairwise and point-wise recommender models, which aims to improve the items sampling in the recommender model that needs negative items in models training/updating.

Finally, I conclude this thesis and present the future works in Chapter 8.

2

Theoretical Background

This chapter presents a revision of related research, seeking to provide a general understanding of the addressed topic, the often-used techniques, and the different strategies to recommendation scenarios. Section 2.1 introduces the learning schemes most often used in recommendation system's researches. Section 2.2 presents the main recommendation approaches. Section 2.3 presents the recommendation challenges related to our research, including the problems of implicit positive-only feedback (Section 2.3.1), concept-drift (Section 2.3.2), and cold-start (Section 2.3.3). Finally, Section 2.4 presents popular recommendation datasets.

2.1 Learning Schemes in the Recommendation Scenarios

Recommender systems suggest the items that appear most likely to be preferred by a particular user based on their historical behavior, actions, and feedback (GUO et al., 2017; RABIU et al., 2020). Modeling users' preferences and interests in recommender systems depends on the existence of training examples. In this sense, we can follow two learning schemes: batch (offline) learning or incremental (online) learning (BABÜROGLU; DURMUSOGLU; DERELI, 2021).

2.1.1 Batch learning

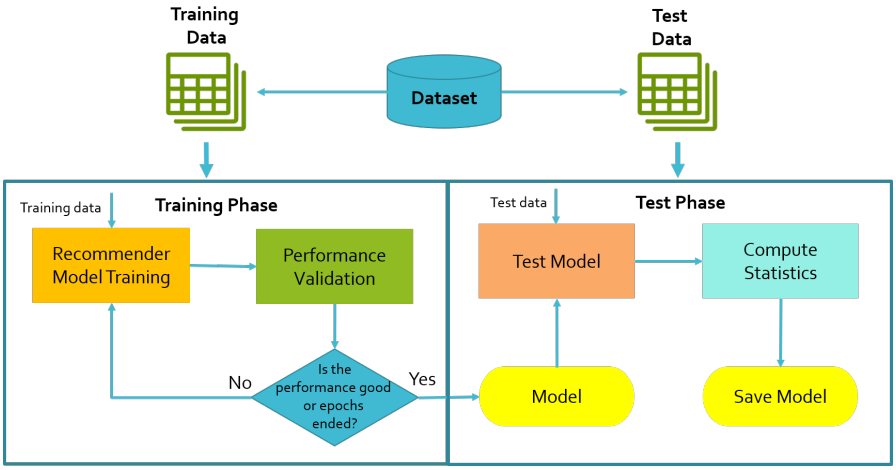
In batch learning, we have to train the recommender model using the entire dataset that is available at a certain point in time (BISONG, 2019). Batch learning algorithms periodically train the recommender models with large-volume historical data, wherein the data are finite, static, and already pre-processed before the recommendation system training. To perform the recommender models training, we can pass through the same dataset multiple times until the model convergence or the fixed number of epochs. Therefore, we can use the generated static model only once the training is completed for performing future predictions (BABÜROGLU; DURMUSOGLU; DERELI, 2021). If the generated model performs well on the test set, we can use the model for production, and thus learning ends. On the other hand, once new data becomes available, we need to update the model based on the new data and train the model from scratch all over again using both the old and new data samples (BISONG, 2019).

The batch learning algorithm is expected to generalize (BURLUTSKIY et al., 2016) in the sense that its output predicts the user's ratings to previously unseen items and also made predictions to users that do not appear in the training set. Therefore, in real-world scenarios, where the data becomes available continuously, and we need to train the model whenever the data arrives, batch learning becomes inappropriate. In such a circumstance, we have to update our learning model on the go, based on the new data samples that are available (BISONG, 2019). Figure 2.1 presents an overview of the batch learning recommendation algorithms. We split the available data into training and test sets. We also select a portion of data for validation from the training set, and we train the model until convergence or training epochs ended. Then, the model generated in the training step is used with the test data to evaluate the model and compute the statistics. Finally, we save the model for future predictions.

2.1.2 Incremental learning

In real-world applications, data are generated continuously at a fast rate in many different domains, such as e-commerce, social media, and digital marketing. These scenarios can be formulated as a data stream problem since the users' interests are dynamic and change over time (RABIU et al., 2020). In this sense, an emerging topic in recommender systems is how to learn from potentially

Figure 2.1: Batch Learning Schema.



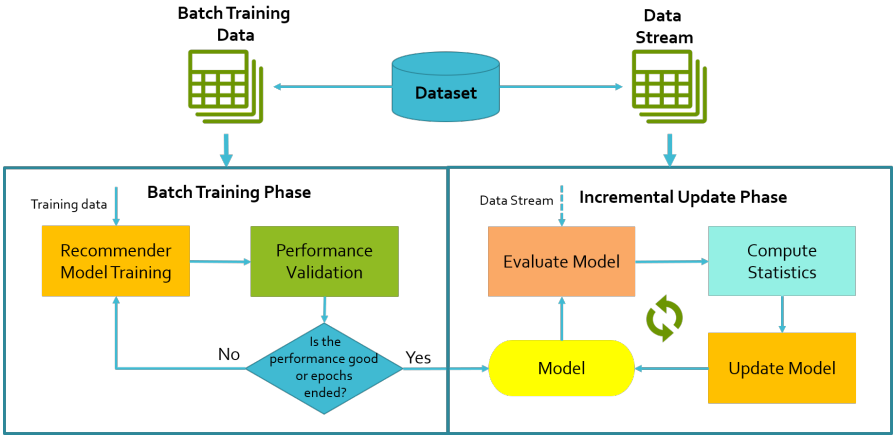
infinite sequences of user-item interactions that arrive over time (VINISKI et al., 2021).

Leaning user's preferences in streaming scenarios remains a challenge due to the users' preferences volatility. The user interests are not static—they undergo changes, i.e., concept drift (MATUSZYK et al., 2015a). Consequently, users' preferences may change over time (concept drift), new items and new users can emerge from the stream (cold start), or even the system conditions can be modified, i.e., sales and actions in specific periods (VINISKI et al., 2021). As a result, the developers of such systems must also be aware of concept drift and cold start issues, which require recommender systems to work incrementally, consistently detecting and adapting to changes in data so that performance is not jeopardized (LAGHMARI; MARSALA; RAMDANI, 2018). Adapting to changes that occur in a changing environment is an essential aspect of every learning system (MATUSZYK et al., 2015a). A feasible way to deal with such a situation is to incorporate new information into a model as new data is made available, using incremental model updating (SONG; CHENG; LU, 2015).

Figure 2.2 presents an incremental learning scheme used in recommendation systems. As in the batch training scheme, we also incorporated a batch training phase into the incremental learning algorithms. We used such a phase to initialize and learn the model weights for the available users and items at the point in time. Thus, we update the generated model in the incremental update phase according to new instances from the data stream. We use the data

stream to evaluate the model, compute statistics, and update the model based on performance.

Figure 2.2: Incremental Learning Schema.



2.2 Recommender Systems

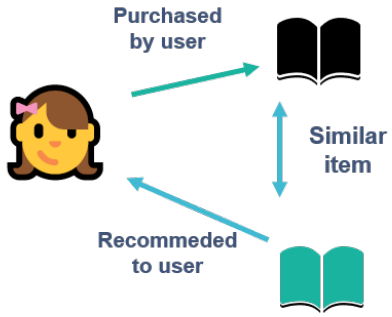
Recommender methods aim at learning and model the user-item relationships in the information systems. Such modeling allows for marketing actions a greater reach, assertiveness, and the definition of customer loyalty strategies through personalized offers. Understanding customer profiles is an essential tool for businesses that wish to stand out in the market and guarantee success in their actions. Thus, recurrently analyzing and identifying changes in users' needs and tastes has proved to be a powerful tool. Besides, the stream mining techniques used in such scenarios gained much attention in recent years. The research community generally divide the recommender systems into three categories: Content-Based Filtering (CB), Collaborative Filtering (CF), and Hybrid Filtering (HF) (ZHANG et al., 2019).

In the following sections, we present the details about the CB, CF, and HF approaches and contextualize the Session-based recommendation process, naturally designed for streaming scenarios with implicit feedback.

2.2.1 Content-Based Filtering

Content-Based filtering techniques, derived from text mining approaches, recommend items to users considering the items characteristics and descriptions (WEI et al., 2017; RICCI; ROKACH; SHAPIRA, 2011). A central component in such methods is the modeling process that make inferences of user interest based on items in which the user previously interacted with. CB models compare the items available in the dataset with items that appear in the user's past interactions and then recommend the best matches (BEEL et al., 2016). Figure 2.3 presents how the recommendation occurs in the content-based approach.

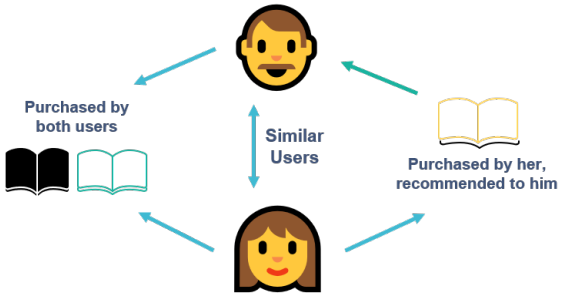
Figure 2.3: Content Based Filtering



2.2.2 Collaborative Filtering

The second approach, Collaborative Filtering, does not need previous knowledge about users or items and is more frequently used in the recommender systems design. Such systems conduct recommendations considering only user-item interactions (NASSAR; JAFAR; RAHHAL, 2020). The rationale behind the collaborative filtering is that users who expressed similar interests will share interests in the future. Thus, recommended items are related to preferred items of users who demonstrated similar interests (PORTUGAL; ALENCAR; COWAN, 2018; YIN et al., 2019). In Figure 2.4, we present the collaborative filtering approach.

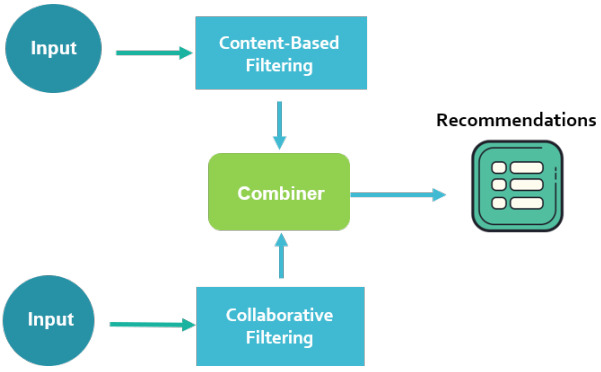
Figure 2.4: Collaborative Filtering



2.2.3 Hybrid Filtering

The hybrid approaches combine both collaborative and content-based techniques to make recommendations. The hybrid recommender models take into account both user-item interaction and the past interacted items characteristics. Thus, such recommender systems can find similarities between users, items, or both (PORTUGAL; ALENCAR; COWAN, 2018). Figure 2.5 represents the recommendations made by a hybrid recommender model.

Figure 2.5: Hybrid Filtering



2.2.4 Session-Based Filtering

Most recommender systems research focuses on models that need the user identifier to build a clear user profile. On the other hand, Session-based Fil-

tering is an emerging recommendation topic and has been widely studied in recent years (TAN; XU; LIU, 2016; LI et al., 2017b; GUO et al., 2019). The session-based recommendation is a typical application of recommender systems based on implicit feedback, where no explicit interests, e.g., ratings; but only positive observations, e.g., clicks; are available. These positive observations are usually sequential data obtained by passively tracking users' behavior over a sequence of time. Compared with traditional user-item settings, this task focuses on sequential mining and can exploit only limited user interactions within a short time frame (GUO et al., 2019). Typically, there are two traditional modeling paradigms, i.e., general recommender and sequential recommender. The general recommender model is related to item-to-item recommendation approaches. An item-to-item similarity matrix is pre-computed from available session-data, and the model then recommends items that often appear together in sessions. Based on Markov chains, the sequential recommender utilizes the sequential data by predicting users next action given the last action (LI et al., 2017b).

2.3 Challenges

In this section we bring forward three challenges related to recommendation systems research area. Section 2.3.1 presents the positive-only feedback problem. Sections 2.3.3 and 2.3.2 introduce cold-start and concept-drift.

2.3.1 Implicit Positive-Only Feedback

The implicit positive-only feedback predicts users' preferences given past positive feedback available in a dataset (LI et al., 2017a). OCCF has characteristics that differentiate it from other tasks in recommendation systems. First, we lack negative feedback, as it is cumbersome to state with certainty which items a user dislikes. For instance, the lack of interaction is ambiguous as the user may dislike an item or be unaware of it. Next, implicit datasets are highly sparse. Few interactions are known, and most of the user-item relationship matrix corresponds to missing/unknown data. Furthermore, OCCF scenarios are noisy, as an interaction between a user and an item does not mean that the user prefers it. There is no explicit feedback from the user w.r.t. one's satisfaction after such interaction. Finally, implicit ratings expressed numerically indicate

confidence and do not represent users' preferences as with explicit ratings, yet, the frequency of interaction, e.g., how many times a user listens to a song, how frequently a user purchases an item, etc. (VINISKI; BARDDAL; BRITTO JR., 2021).

Existing solutions for OCCF differ in terms of how they treat unobserved data. Although the one-class collaborative filtering is less visited than the multi-class setting, some approaches have been proposed in the literature to deal with missing (unknown) items (SONG et al., 2018). According to how the unlabeled data is used, existing methods for OCCF can be classified into two categories (SONG et al., 2018), i.e., whole-data based approaches (PAN et al., 2008; YU; BILENKO; LIN, 2017); and sampling based approaches (RENDLE et al., 2012; ZHANG et al., 2013; HE et al., 2017; YU; BILENKO; LIN, 2017). Both approaches share challenges. Considering all the missing entries as negative, we have to deal with two limitations. First, as most training instances are negative, the class imbalance problem reduces the positive class's predictive ability. Second, we have to deal with the possibility of introducing false negative examples. Besides, suppose we randomly sample unobserved interactions. In that case, it is challenging to identify representative negative examples as all of the negative and missing positive interactions are mixed and cannot be distinguished (PAN et al., 2008). On the other hand, if the sampling method considers the dataset characteristics during negative sampling, we have less probability of selecting false negatives in the training stage (VINISKI; BARDDAL; BRITTO JR., 2021).

2.3.2 Concept Drift

An emerging topic in recommender systems is how to learn from potentially infinite sequences of user-item interactions that arrive over time. The building of models that learn incrementally from continuous flows of data is a concern of data stream mining (AGGARWAL, 2007; GAMA, 2010). Models tailored for data streams must have low computational costs regarding processing time and memory consumption while also tackle concept drifts. Concept drift is a problem that arises when the data distribution changes, thus affecting its underlying patterns (TSYMBAL, 2004; WEBB et al., 2018). If a model cannot detect and adapt to drifts swiftly, prediction power is put in jeopardy.

In recommender systems, concept drift refers to changes in customer behavior, which reflect changes in their preferences (JORGE et al., 2016). Consequently,

recommender systems should be aware that the relationship between users and items is not static. Thus, these are expected to detect and adapt to changes accordingly (BABÜROGLU; DURMUSOGLU; DERELI, 2021; GAMA et al., 2014).

The most efficient way to deal with potentially drifting scenarios is to increment the model as user-item interactions are available. Recommender systems that are tailored to handle data streams must address both the problem of learning drifting behavior and computational complexity issues (VINISKI et al., 2021).

2.3.3 Cold-Start

Most collaborative filtering approaches require a large number of ratings from a user on an item to provide valuable recommendations, thus leading to unreliable suggestions due to an initial absence of rating (OCEPEK; RUGELJ; BOSNIC, 2015). Although collaborative filtering is widely followed in recommender systems' implementation, its techniques suffer from sparsity, and cold start problems (WEI et al., 2017). Data sparsity occurs when the number of interactions between users and items is much smaller than the total number of user and item combinations. Consequently, the mapping between the latent factors and the rating matrix R becomes even more cumbersome (SHAO; LI; BIAN, 2021).

On the other hand, the cold start problem occurs when either a new user or an item appears over time. Thus, the recommender model cannot map the user-item interactions accurately as no *a priori* information on either of them is available. In recommender systems, the cold start problem includes three cases: (1) cold start of users (how to recommend items to a user recently entered the system); (2) cold start of items (how to recommend a new item recently introduced into the system to interested users); and (3) cold start of the system (how to make accurate recommendation in a new system) (PANDEY; RAJPOOT, 2016; WEI et al., 2017).

Several techniques have been proposed in the literature to solve the cold start problem and deal with data sparseness (PANDEY; RAJPOOT, 2016; WEI et al., 2017; SILVA et al., 2019; TAHMASEBI et al., 2020; BI et al., 2020). However, most of these techniques require additional data, such as demographic and social network information. Nonetheless, in real-world scenarios, additional information is often unavailable, and the system has to deal with the provided data to alle-

viate these problems. A recent approach for addressing cold-start scenarios is using data stream methods, as incremental approaches are continuously learning the new underlying patterns between users and items (CHANDRAMOULI et al., 2011; JOSÉ; ENEMBRECK; BARDDAL, 2020).

2.4 Datasets

Most of the available datasets in the literature are related to movies, songs, books, and articles. A small group of datasets represents a large proportion of research papers in the recommender systems community. The *Movie Lens* (HARPER; KONSTAN, 2016) is one of the most used datasets, and by providing many user ratings and being updated continuously has been used in the testing and development of very collaborative recommendation algorithms. The *Yahoo Movie Ratings*, *Netflix Prize* and *Movie Tweetings* movie datasets are also widely adopted to improving recommendations systems researches (ÇANO; MORISIO, 2015). In addition to the movies' data and the popularity of video stream platforms, the music recommendation algorithms are also popular in the research area. *Last.fm* and *Yahoo Music Ratings* were used many times in model comparison, as well as the algorithms' development. Despite providing support to improve the research area, most of these recommendation datasets were sufficiently studied and did not provide new challenges related to the recommendation process.

In addition to being frequently used in research related to recommender systems, most of these popular datasets present explicit feedback, i.e., these datasets contain the user ratings to items. However, many pieces of research that work with implicit (positive only) feedback transform these datasets into implicit form, sometimes using only the high score (5 stars, for example) as an event (interaction between user and item) or using all ratings as positive events (PAN et al., 2008). Although the advantage of explicit feedback is clear, considering the facility to recover user preferences, the implicit feedback is easier to obtain, because in most of the cases does not need adaptation in websites for rating collection (PAN; LIU; MING, 2016). Besides that, a user does not rate every song he/she listens to or the movies watched. Therefore, we do not have all the information about users' events when transforming an explicit dataset into an implicit one. Thereby, making datasets with naturally implicit feedback available has very importance for research involving recommendation. The studies with

implicit feedback have become popular, and the number of research articles on implicit feedback has increased in recent years. In particular, a large group of datasets has been mainly shared across competitions like NetFlix, Kaggle, or RecSys (SIDANA et al., 2017).

Considering the retail scenario, which has processes similar to supermarkets, few data collections are explicitly designed for recommendations, and most of the existing retail datasets were created for market basket analysis and collected from online purchases. Some existing online supermarket datasets available in the literature and most frequently used for recommendations are *Instacart Dataset from Kaggle* [1], *Groceries Dataset* [2], *Tafeng*[3], and *BeiRen*[4].

In the work of (TATIANA; MIKHAIL, 2018), *Instacart Dataset* and *Groceries Dataset* are used for improving recommendations using association rules mining. The *TaFeng* and *BeiRen* are two online shopping datasets with real purchase records of users and have been used together for many researchers for recommendation (WANG; GUO; LAN, 2014; WANG et al., 2015; LE; LAUW; FANG, 2017; BAI et al., 2019). According to (BAI et al., 2019), these datasets are the only publicly available ones we are aware of that contain the real purchase history of users. *BeiRen* dataset is collected by a large retail department store in China, which records its supermarket purchase history during the period from January 2011 to September 2013. It contains 1,123,754 transactions belonging to 34,221 users and 17,920 items. *Tafeng* dataset is a public dataset offered by RecSys Conference, which covers products from food, office supplies to furniture. It contains 817,741 transactions belonging to 32,266 users and 23,812 items.

All presented supermarket datasets have been preprocessed in the recommendation experiments due to the high data sparsity and because most users and items appear a few times in the dataset. Similarly, in the works of (WANG; GUO; LAN, 2014; WANG et al., 2015; LE; LAUW; FANG, 2017), the authors conduct some preprocessing on the datasets transaction. For both TaFeng and BeiRen datasets, all the items bought by less than ten users were removed , and users that have bought less than ten items as well. When these preprocessing steps are made, only 9,238 and 9,321 users, 7,982 and 5845 items, and 67,964 and

[1] https://www.kaggle.com/c/instacart-market-basket-analysis
[2] https://www.kaggle.com/datasets/heeraldedhia/groceries-dataset
[3] https://www.kaggle.com/datasets/chiranjivdas09/ta-feng-grocery-dataset
[4] http://www.brjt.cn

91,294 transactions are used in *Tafeng* and *BeiRen* datasets, respectively. Such preprocessing means that more than 50% of the transactions were filtered, and even though these datasets possess many transactions, few are actually assessed.

For the retail area, in addition to making good recommendations and modeling users' preferences/confidence, recommender systems can be beneficial to find best practices, assist decision making, and provide support to generate marketing actions (BALAKRISHNAN et al., 2018). Recently, e-commerce has become an essential channel for many retail businesses, and recommender systems have been widely used as an innovative solution that overcomes the limitations of e-commerce services (HWANGBO; KIM; CHA, 2018).

Some considerations about retail recommendations are related to the data source. Online and physical shopping do not share the same characteristics, and users who buy from the online store do not necessarily buy from the physical store. On the other hand, most of the available datasets are obtained in online shopping, and recommendations systems for this kind of data do not take into account the physical store events (ZAKARIA et al., 2014). Historically, it is not easy for the physical supermarkets to obtain information about specific users' consumption patterns, since most stores did not have customer registration processes. For these stores, it was infeasible to incorporate personalized actions and offer specific products for specific users without user purchase information. However, with the popularization of advantage clubs, loyalty cards, and the possibility to link the personal code in the commercial invoice, its also possible to obtain the information about specific customers' consumption in the physical stores (GÓMEZ; ARRANZ; CILLÁN, 2012).

Furthermore, the retail market has some recurrent characteristics that encourage incremental and adaptive recommendation methods. For example, supermarkets offer different products on specific days of the week. Moreover, seasonality has a significant influence on user-item events. For example, some products are purchased more frequently on the weekend, e.g. beers and soft drinks, and holidays increase the consumption of other products, e.g. panettone and turkey at Christmas.

Therefore, the physical supermarket data availability can be helpful to increase the recommendation research in the retail area, and it is a great scenario to analyze temporal aspects in recommender systems. In this sense, in Chapter 5 we introduce the supermarket dataset (SMDI) obtained from the supermarket purchases (VINISKI et al., 2021). We also propose the challenge of modeling

users' preferences considering only four months of supermarket transactions. In addition, there are no many public supermarket datasets available, and few recommendation algorithms are designed specifically for supermarket recommendations. As for other publicly available data collections, the primary purpose of the proposed dataset is to encourage research on recommender systems algorithms for retail and to provide a reference based on implicit feedback for evaluation (SIDANA et al., 2017).

2.5 Final Considerations

This chapter presented the recommender systems' theoretical background regarding learning schemes (batch and stream), recommendation approaches (content-based, collaborative, and hybrid filtering), challenges (concept drifts, cold-start, data sparsity, and negative sampling), and popular available datasets. This thesis envisions testing batch and stream recommender models and providing an experimental protocol for comparison. The Collaborative Filtering recommendation approach is primarily focused due to the ease of collecting data in collaborative scenarios and the challenges in modeling user-item relationships. According to the recommendation challenges, this thesis implicitly focused on concept drifts and cold-start problems in streaming scenarios and explicitly generated alternatives to improving the negative item selection. Next, Chapter 3 presents popular collaborative recommendation models for batch and streaming learning schemes.

3

Related Works

This chapter presents an overview of existing works related to implicit positive-only recommender models used as baseline models to develop streaming recommendation systems.

Thus, Section 3.1 presents the popular batch techniques often used in positive-only feedback scenarios, and Section 3.2 brings forward the existing works in the streaming recommendation area.

3.1 Batch Recommendation Models for OCCF

This section reviews some popular approaches designed for OCCF scenarios that sample negative examples for training recommender models. We can categorize these approaches according to how they learn the relevance order. Most algorithms exploiting OCCF focus on homogeneous positive feedback with point-wise (HE et al., 2017), pair-wise (RENDLE et al., 2012), and list-wise (SHI; LARSON; HANJALIC, 2010) preference assumptions. Point-wise approaches regard user ratings as categorical labels or numerical values and learn the relevance scores of missing data directly (SONG et al., 2018). The pairwise approaches try to capture the preference order between missing data, correctly identifying the positive/negative item in each pair (ZHANG et al., 2013; SONG et al., 2018). On the other hand, an individual training example is an entire list of items in a list-wise approach, rather than individual items or item pairs. However, due to their difficulty modeling the inter-list loss and

inefficiency on large scale datasets, list-wise CF approaches are not widely used compared to point-wise and pairwise in ranking-oriented collaborative filtering (ZHANG et al., 2013).

Consequently, for further experimentation, we select recommender models based on Matrix Factorization, such as Singular Value Decomposition (SVD) (KOREN; BELL; VOLINSKY, 2009), Bayesian Personalized Ranking for Matrix Factorization (BPRMF) (RENDLE et al., 2012), Alternating Least Squares (ALS) (CHEN et al., 2020), the k-nearest neighbors (kNN) - Neighborhood Model, and the models proposed in the Neural Collaborative Framework (NCF): Generalized Matrix Factorization (GMF), Multi-layer Perceptron recommender (MLP) and Neural Matrix Factorization (NeuMF) (HE et al., 2017).

3.1.1 Matrix Factorization Models

MF is the most commonly used technique in recommendation system design (TAKÁCS et al., 2009; YU et al., 2016), and the most successful latent factor models are also based on MF (KOREN; BELL; VOLINSKY, 2009). MF models have been widely applied to different recommendation scenarios and outperform clustering and neighborhood methods in terms of predictive power, run time (VINAGRE; JORGE; GAMA, 2014), and scalability (KOREN; BELL; VOLINSKY, 2009).

The input of MF is a relation matrix between users and items $R \in \mathbb{R}^{m \times n}$, where m denotes the number of users and n denotes the number of items (CHEN et al., 2020). Because of R's sparsity, MF models map users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space (KOREN; BELL; VOLINSKY, 2009; CHEN et al., 2020). The number of latent factors (f) is much smaller than the number of users and items, and the co-occurrence between users and items forms the basis for recommendations (TAKÁCS et al., 2009).

More formally, matrices $R \in \mathbb{R}^{m \times n}$, $\overrightarrow{A} \in \mathbb{R}^{m \times f}$ and $\overrightarrow{B} \in \mathbb{R}^{f \times n}$ represent users' ratings to items, users' latent vectors, and items' latent vectors, respectively. The $r_{u,i}$ entry in the u-th row and i-th column of R is the rating that user u gives to item i. u-th row vector (p_u) of $\overrightarrow{A}$ and i-th column vector (q_i) of $\overrightarrow{B}$ are the user's u and item's i latent vectors, respectively (YU et al., 2016). Given this formulation, it is possible to compute predicted rating of user u for the i-th item using the

dot product depicted in Equation 3.1 (VINAGRE; JORGE; GAMA, 2014).

$$\widehat{r}_{ui} = \overrightarrow{A_u} \cdot \overrightarrow{B_i}^T \tag{3.1}$$

MF is closely related to singular value decomposition (SVD), a well-established technique for information retrieval to identify latent features (RAGHUWANSHI; PATERIYA, 2018). Applying SVD to collaborative filtering requires factoring in the user-item matrix. However, the conventional SVD is undefined when the rating matrix is incomplete. Because of the high percentage of missing values, some methods use imputation techniques to fill in the missing values and make the rating matrix dense before applying SVD. However, these approaches can be costly and considerably distort the data owing to inaccurate imputation (KOREN; BELL; VOLINSKY, 2009; RAGHUWANSHI; PATERIYA, 2018). Hence, more recent researches suggest direct modeling of the observed ratings while avoiding overfitting through a regularized model (KOREN; BELL; VOLINSKY, 2009; RAGHUWANSHI; PATERIYA, 2018). Thus, the model was trained by minimizing L2-regularized squared error for known values of $\widehat{R}$ and the corresponding predicted ratings are denoted in Equation 3.2 (VINAGRE; JORGE; GAMA, 2014).

$$\min_{\overrightarrow{A},\overrightarrow{B}} \sum_{(u,i)\in D} \left(r_{u,i} - \overrightarrow{A_u} \cdot \overrightarrow{B_i}^T\right)^2 + \lambda \left(\left\|\overrightarrow{A_u}\right\|^2 + \left\|\overrightarrow{B_i}\right\|^2\right) \tag{3.2}$$

In the formalization above, D is the set of (u, i) pairs for each known $r_{u,i}$ (training set), and λ is the regularization parameter for user $\overrightarrow{A_u}$ and item $\overrightarrow{B_i}$ latent vectors that are used to avoid overfitting. MF learns a model by fitting previously observed interactions. However, the goal is to generalize previously known user-item interactions to predict future, unobserved user preferences over items (KOREN; BELL; VOLINSKY, 2009). Approaches to minimizing Equation 3.2 include Stochastic Gradient Descent (SGD) (KOREN; BELL; VOLINSKY, 2009), Alternating least Squares (ALS) (CHEN et al., 2020), Bayesian Personalized Ranking (BPR) - an adaptation of the SGD technique for pairwise learning Rendle et al. (2012). The following sections show the matrix factorization optimizers (SGD, ALS, BPR).

Stochastic Gradient Descent (SVD++)

The most common approach for minimizing Equation 3.2 in MF is the SGD optimization, in which the algorithm loops over all ratings in the training set. For each given training interaction, the system predicts r_{ui} and computes the associated prediction error using Equation 3.3 (KOREN; BELL; VOLINSKY, 2009).

$$err_{ui} = r_{ui} - \widehat{r}_{ui} \tag{3.3}$$

In this process, both the user ($\overrightarrow{g_u}$) and item ($\overrightarrow{g_i}$) gradients of the error (Equation 3.4) are used, where λ is the regularization rate.

$$\begin{aligned} \overrightarrow{g_u} &\leftarrow err_{ui} \times \overrightarrow{B_i} - \lambda \overrightarrow{A_u} \\ \overrightarrow{g_i} &\leftarrow err_{ui} \times \overrightarrow{A_u} - \lambda \overrightarrow{B_i} \end{aligned} \tag{3.4}$$

Next, SGD updates the parameters by a magnitude proportional to η in the inverse direction of the error's gradient according to Equation 3.5 (KOREN; BELL; VOLINSKY, 2009), where η is the step size or learning rate, and λ is the regularization term for both user and item latent factors (VINAGRE; JORGE; GAMA, 2014).

$$\begin{aligned} \overrightarrow{A_u} &\leftarrow \overrightarrow{A_u} + \eta \times \overrightarrow{g_u} \\ \overrightarrow{B_i} &\leftarrow \overrightarrow{B_i} + \eta \times \overrightarrow{g_i} \end{aligned} \tag{3.5}$$

Alternating Least Squares

Alternating least squares (ALS) is an efficient MF technique for recommender systems (CHEN et al., 2020). Because both A_u and B_i are unknowns, Equation 3.2 is not convex. However, the minimization principle of ALS is to keep one fixed while calculating the other. Thus, the ALS technique fixes the B matrix to calculate the A matrix to get vectors A_u, and vice-versa. Consequently, the optimization problem becomes a quadratic function and can be solved optimally. The procedure ensures that each step decreases Equation 3.2 until convergence. First, we minimize the equation over A while fixing B, and we obtain Equation

3.6 (KOREN; BELL; VOLINSKY, 2009; CHEN et al., 2020).

$$\min_{A} \sum_{(i)\in D_u} \left(r_{u,i} - A_u B_i^T \right)^2 + \lambda(\|A_u\|^2) \tag{3.6}$$

By calculating the partial derivative of A_u in Equation 3.6 and letting the partial derivative equal to zero, we obtain the Equation 3.7.

$$A_u = (B^T B + \lambda I)^{-1} B^T r_u, \tag{3.7}$$

where I is the unit matrix ranked f, and r_u is the u-th rows of R. Similarly, we obtain B_i in Equation 3.8.

$$B_i = (A^T A + \lambda I)^{-1} A^T r_i. \tag{3.8}$$

Bayesian Personalized Ranking for Matrix Factorization

Bayesian Personalized Ranking (BPR) for MF is a generic optimization criterion for personalized ranking derived from a Bayesian analysis of the problem. Rendle et al. (2012) provide a learning BPR method based on SGD with bootstrap sampling (BREIMAN, 1996). In addition to the matrix factorization parameters, for each instance $(u; i)$ in the training set, BPRMF selects a negative item j (an item that the u-th user did not interact with). BPR optimization decomposes triplets in the $(u; i; j)$ format using the difference of the u-th user predictions w.r.t. items i and j, such as depicted in Equation 3.9.

$$\widehat{r}_{uij} = \widehat{r}_{ui} - \widehat{r}_{uj} \tag{3.9}$$

Next, the model applies a sigmoid function variant described in Equation 3.10 (DING et al., 2018) to the prediction $\widehat{r}_{uij}$ to add the Bayesian probabilistic characteristic to the model.

$$\sigma(\widehat{r}_{uij}) = \left(\frac{1}{1 + e^{-\widehat{r}_{uij}}} \right) \tag{3.10}$$

For each triplet $(u; i; j)$, the latent factor vectors for a user u, item i, and unobserved item j are updated using Equation 3.11.

$$\Theta \leftarrow \Theta + \eta \left(\sigma(\widehat{r}_{uij}) \times \frac{\partial}{\partial_\Theta} \widehat{r}_{uij} - \lambda\Theta \right) \tag{3.11}$$

where

$$\sigma(\widehat{r}_{uij}) \times \frac{\partial}{\partial_\Theta} \widehat{r}_{uij} = \begin{cases} (\overrightarrow{B_i} - \overrightarrow{B_j}) & \text{if } \Theta = \overrightarrow{A_u} \\ \overrightarrow{A_u} & \text{if } \Theta = \overrightarrow{B_i} \\ -\overrightarrow{A_u} & \text{if } \Theta = \overrightarrow{B_j} \\ 0 & \text{otherwise} \end{cases} \tag{3.12}$$

As the BPRMF has an additional term j, the model differs in SGD to calculate the gradients of the error for each term in the triplet $(u; i; j)$ (Equation 3.13).

$$\begin{aligned} \overrightarrow{g_u} &\leftarrow \sigma(\widehat{r}_{uij}) \times (B_i - B_j) - \lambda \overrightarrow{A_u} \\ \overrightarrow{g_i} &\leftarrow \sigma(\widehat{r}_{uij}) \times (A_u) - \lambda \overrightarrow{B_i} \\ \overrightarrow{g_j} &\leftarrow \sigma(\widehat{r}_{uij}) \times (-A_u) - \lambda \overrightarrow{B_j} \end{aligned} \tag{3.13}$$

Finally, decomposing Equations 3.11 and 3.12 for each component of the $(u; i; j)$ instance, the latent factor vectors for user u, item i, and unobserved item j are updated using the traditional SGD optimization strategy described in Equation 3.14.

$$\begin{aligned} A_u &\leftarrow A_u + \eta \times \overrightarrow{g_u} \\ B_i &\leftarrow B_i + \eta \times \overrightarrow{g_i} \\ B_j &\leftarrow B_j + \eta \times \overrightarrow{g_j} \end{aligned} \tag{3.14}$$

Momentum-incorporated Latent Factor Model

The Momentum-incorporated latent factor (MLF) algorithm is an optimization strategy introduced as part of the momentum-incorporated parallel SGD (MPSGD) model (LUO et al., 2021). The main purpose of the MPSGD model is to paralyze the update of latent factors during the batch training step of the MF model (LUO et al., 2021). In contrast, MLF is an SGD-based learning method designed to update the latent factors matrices of user A and item B, which can be integrated into an incremental learning scenario.

The momentum method is an adaptive optimization strategy for improving the convergence rate of an SGD-based model. The momentum algorithm improves the model convergence by adding a fraction γ of the previous interaction updates to the current interaction update. Thus, the learning update in the current interaction considers the direction trend from the previous updates, reducing oscillations during the learning process and making the resultant model

converge faster (GREENBERG-TOLEDO et al., 2019; LUO et al., 2021). Equation 3.15 presents the update rules of the momentum optimizer, where $\overrightarrow{v_t}$ stores the history of the previous update, γ is the momentum term (also known as the decay rate), $\overrightarrow{g}$ represents the gradients of the error, and $\overrightarrow{\theta}$ represents the latent factor vector (A_u, B_i).

$$\begin{aligned} \overrightarrow{v_t} &\leftarrow \gamma \times \overrightarrow{v_{t-1}} + \eta \times \overrightarrow{g} \\ \overrightarrow{\theta} &\leftarrow \overrightarrow{\theta} - \overrightarrow{v_t} \end{aligned} \tag{3.15}$$

The MLF incorporates momentum optimization by introducing $V_{(A)}$ and $V_{(B)}$ auxiliary arrays. These arrays store the history of previous updates for each user u and item i in the dataset. Consequently, to update the user latent factor vector $\overrightarrow{A_u}$ with the MLF model, the update rules depicted in Equation 3.16 are used.

$$\begin{aligned} \overrightarrow{v_{u(t)}} &\leftarrow \gamma \times \overrightarrow{v_{u(t-1)}} + \eta \times \overrightarrow{g_u} \\ \overrightarrow{A_u} &\leftarrow \overrightarrow{A_u} - \overrightarrow{v_{u(t)}} \end{aligned} \tag{3.16}$$

Similarly, for the item latent factor vector $\overrightarrow{B_i}$ updates, we used the rules in Equation 3.17.

$$\begin{aligned} \overrightarrow{v_{i(t)}} &\leftarrow \gamma \times \overrightarrow{v_{i(t-1)}} + \eta \times \overrightarrow{g_i} \\ \overrightarrow{B_i} &\leftarrow \overrightarrow{B_i} - \overrightarrow{v_{i(t)}} \end{aligned} \tag{3.17}$$

3.1.2 Neighborhood Models

The neighborhood models (NBMs) are common approaches used for collaborative filtering in recommendations systems terms. NBMs are traditionally sub-categorized into user-based models (UBMs) (HERLOCKER et al., 1999) and item-based models (IBMs) (SARWAR et al., 2001) according to the target of the similarity measures (CHAE et al., 2018). Thus, the neighborhood methods are centered on producing predictions $\widehat{r}_{ui}$ based on the set of k users $\mathcal{N}_u$ most similar to u (i.e., UBMs) or, alternatively, the set of k items $\mathcal{N}_i$ most similar to i (i.e., IBMs) (KOREN, 2008). Considering the set of m users $\mathcal{U}$ and the set of n items $\mathcal{I}$, the set of items a user u interacted with ($\mathcal{I}_u$) and the set of items a user v interacted with ($\mathcal{I}_v$), to obtain the prediction $\widehat{r}_{ui}$, it is necessary to compute the pairwise similarity between users, measured by the cosine similarity (Equation

3.18).

$$S_{uv} = \frac{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in \mathcal{I}_u} r_{ui}^2} \cdot \sqrt{\sum_{i \in \mathcal{I}_v} r_{vi}^2}} \tag{3.18}$$

where the item-item similarity S_{ij} can be defined symmetrically based on $\mathcal{U}_i$ and $\mathcal{U}_j$. Based on these similarities, we could then determine the neighborhood $\mathcal{N}_u$ and $\mathcal{N}_i$ for each user u or item i consisting of the k users or items closest to u or i. The predicted score $\widehat{r}_{ui}$ is then determined by the proportion of observed ratings in $\mathcal{N}_u$ or $\mathcal{N}_i$. In the user-based model, we obtain the predictions as in Equation 3.19 (LIU et al., 2010).

$$\widehat{r}_{ui} = \frac{\sum_{v \in \mathcal{N}_u \cap \mathcal{U}_i} S_{uv}}{\sqrt{\sum_{v \in \mathcal{N}_u} S_{uv}}} \tag{3.19}$$

the reason for this interpretation lies in the fact that an item will be appear in a future user interaction if more neighbors have interacted with it.

3.1.3 Neural Collaborative Framework - NCF

NCF is a deep neural network recommender framework composed of three recommender models: GMF, MLP, and NeuMF (HE et al., 2017). The NCF framework presents a probabilistic approach for learning the point-wise models that pay special attention to implicit data's binary property, i.e., training models using positive and negative examples. To endow the probabilistic explanation, NCF models constrain the output $\hat{r}_{ui}$ in the range of [0, 1] using a probabilistic function in the output layer. Regarding negative instances, the authors suggest uniformly sampling them from unobserved interactions in each iteration.

Generalized Matrix Factorization

The Generalized Matrix Factorization (GMF) model results from an extensive literature investigation of factorization models. It is a particular case of the Neural Collaborative Framework (NCF) (HE et al., 2017). As input, GMF receives the one-hot encoded feature vectors v_u^U and v_i^I that describe user u and item i, respectively. Above the input layer are the embedding layers $A_u = A^T v_u^U$ and $B_u = B^T v_i^I$, which are the latent vectors of user and item. NCF's first layer mapping function is given by Equation 3.20, where $\odot$ denotes the element-wise

product of user and item latent vectors.

$$\phi(A_u, B_i) = (A_u \odot B_i) \tag{3.20}$$

Each embedding layer (latent vector) is a fully connected layer that projects the sparse representation of users and items to a dense vector. Thus, projecting the vector to the output layer, the rating prediction of the u-th concerning the i-th item is obtained, as denoted in Equation 3.21, where a_{out} and h represent the activation function and edge weights of the output layer, respectively.

$$\widehat{r}_{ui} = a_{out}\left(h^T\left(A_u \odot B_i\right)\right), \tag{3.21}$$

Finally, the matrix factorization model is computed using an identity function for a_{out} and enforcing h to being a uniform vector of ones (HE et al., 2017).

Multi-layer Perceptron (MLP)

Similar to GMF, the multi-layer perceptron (MLP) for recommender systems is also part of the NCF framework (HE et al., 2017). The first difference between GMF and MLP resides in the first layer. MLP concatenates the user and item latent features instead of using the element-wise dot product between latent factors as in GMF. Second, the model has hidden layers on the concatenated vector to model the collaborative filtering effect and learn the interaction between Au and Bu latent features. More formally, Equation 3.22 describes the MLP model, where W_x, b_x, and a_x denote the weight matrix, bias vector, and activation function for the x-th layer, respectively.

$$\begin{aligned} z_1 = \phi_1(A_u, B_i) &= \begin{bmatrix} A_u \\ B_i \end{bmatrix}, \\ \phi_2\left(z_1\right) &= a_2\left(W_2^T z_1 + b_2\right), \\ &\cdots \\ \phi_L(z_{L-1}) &= a_L(W_L^T z_L + b_L), \\ \widehat{r}_{ui} &= \sigma\left(h^T \phi_L\left(z_{L-1}\right)\right), \end{aligned} \tag{3.22}$$

Neural Matrix Factorization

Neural Matrix Factorization (NeuMF) combines GMF and MLP architectures. More specifically, it combines the linear kernel from GMF and the non-linear kernel from MLP. Internally, NeuMF trains GMF and MLP with random initializers until convergence. NeuMF allows GMF and MLP to learn separated embeddings and merge them by concatenating their last hidden layer to provide more flexibility to the combined model. A formal description for NeuMF is given in Equation 3.23 (HE et al., 2017), where A_u^G and A_u^M denote the user embedding for GMF and MLP parts, respectively, while similar notations of B_i^G and B_i^M hold for item embeddings.

$$\begin{aligned} \phi^{GMF} &= A_u^G \odot B_i^G, \\ \phi^{MLP} &= a_L\left(W_L^T\left(a_{L-1}\left(\ldots a_2\left(W_2^T\begin{bmatrix}A_u^M\\B_i^M\end{bmatrix}+b_2\right)\ldots\right)\right)+b_L\right), \\ \widehat{r}_{ui} &= \sigma\left(h^T\left(\begin{bmatrix}\phi^{GMF}\\\phi^{MLP}\end{bmatrix}\right)\right), \end{aligned} \tag{3.23}$$

3.2 Streaming Recommendation Models

In real-world applications, users' feedback is made available continuously. Besides, we have no control over the order in which data arrives. Let us suppose that new information becomes available in batch modes, such as a new item, a new user, or a new user-item event. In that case, it is necessary to retrain the entire matrix factorization (MF) model using both old and new data. The retraining process may be unfeasible since batch training is computationally expensive (YU et al., 2016; WU; WANG; CHENG, 2008). Furthermore, there is no guarantee that the past user-item relationships are consistent with those observed in more recent data. Consequently, we should update the recommender systems in a single-pass manner according to the arrival of user-item interactions, instead of retraining the entire model (VINAGRE; JORGE; GAMA, 2014). As a result, the practical benefits of using single-pass incremental systems encompass computational cost reduction (WU; WANG; CHENG, 2008) and drift adaptation (MATUSZYK et al., 2015a; CHANG et al., 2017).

Several topics are closely related to stream learning approaches in recommender systems, such as Temporal Dynamic, Adaptive Learning, Incremen-

tal Learning, Online Learning, and Real-Time Learning. Most of the existing streaming recommenders are related to the collaborative filtering approach and use matrix factorization methods to learn and model user preferences (RENDLE et al., 2012; LUO; XIA; ZHU, 2012; VINAGRE; JORGE; GAMA, 2014; YU et al., 2016). We can also see temporal clustering (RANA; JAIN, 2014; ZHOU et al., 2020) and temporal association rules (ZHOU; HIRASAWA, 2019) to generate time-dependent recommendations to users, categorized as Hybrid approaches. According to the feedback type, explicit and implicit feedback are used to model user-item relationships and provide valuable suggestions to the system's users. Thus, the following sections provide the results of a search in the related fields to cover the main approaches designed for Streaming Recommendations' research area.

3.3 Incremental Learning

This Section presents the popular Incremental Learning Recommender algorithms designed for streaming scenarios.

3.3.1 Incremental Stochastic Gradient Descent (ISGD)

The Incremental Stochastic Gradient Descent (ISGD) (VINAGRE; JORGE; GAMA, 2014) is a recommender model designed initially for positive-only feedback recommendations. Therefore, we represent the interactions between users and items in the boolean matrix R, where $R_{u,i} = 1$ stands for the existence of an interaction between the u-th user and the i-th item, and $R_{u,i} = 0$ depicts the absence of such relation. As in the traditional SVD++ model, we predict the rating ($\widehat{r}_{ui}$) of a user u to an item i by the dot product of their respective latent factor vectors Au and B_i^T (see Equation 3.1). On the other hand, as the ISGD model is designed for item prediction and the model outputs is a numerical value, we can select the most relevant items to a user in two ways: the highest predicted values or the closest to 1. Furthermore, we must calculate the prediction error as in Equation 3.24.

$$err_{ui} = 1 - \widehat{r}_{ui} \tag{3.24}$$

Similar to the traditional batch MF model, the ISGD then updates the A_u

and B_i vectors using the Equation 3.5. A relevant difference between ISGD and its batch counterpart (SVD MF with batch SGD optimizer) regards the order in which the method analyzes the data to generate the models (VINISKI et al., 2021). The traditional batch SVD shuffles and performs multiple passes over the training data during pre-processing and model creation. Differently, ISGD does not perform any data pre-processing and processes data according to their natural arrival order (VINAGRE; JORGE; GAMA, 2014).

3.3.2 Incremental Bayesian Personalized Ranking for Matrix Factorization (IBPRMF)

The Incremental Bayesian Personalized Ranking for Matrix Factorization (IBPRMF) updates vectors A_u, B_i and B_j using the operations present in Equations 3.11 and 3.12 as each user-item interaction of the dataset is made available (RENDLE et al., 2012).

In this sense, similar to ISGD, we do not generate a static model to be used later for predictions. The IBPRMF recommender continuously updates the model parameters according to new instances emerging from the stream. Another significant difference between IBPRMF and BPRMF resides in the set of negative items available for selection during a user-item interaction model update. In the streaming variant (IBPRMF), the negative item is selected based on items that appeared thus far in the stream and did not interact with the current user (RENDLE et al., 2012).

Probabilistic Matrix Factorization

Similarly to the traditional MF model, the Probabilistic MF (PMF) model decomposes a binary matrix into two lower-rank matrices to make predictions (SALAKHUTDINOV; MNIH, 2007). The main difference is related to the use of positive (ones, 1) and unknown (zeros, 0) observations during model training, that is, the matrix represents a user's likes and potential dislikes for items. PMF is also similar to the GMF model, which is part of the NCF framework (Section 3.1.3). However, GMF uses neural network structures (embedding) to represent latent factor vectors and was designed for batch processing schemes. Equation 3.25 describes the PMF process, where σ represents a sigmoid function (σ)

applied to make probabilistic predictions ($0 \leqslant \widehat{r_{ui}} \leqslant 1$).

$$\widehat{r_{ui}} \leftarrow \sigma\left(\overrightarrow{A_u} \cdot \overrightarrow{B_i^T}\right) \tag{3.25}$$

In the update process of the PMF's user ($\overrightarrow{A_u}$) and item ($\overrightarrow{B_i}$) latent factor vectors, ISGD update rules are used (Equation 3.5), except for error calculation, because the PMF uses both positive and negative (unknown) observations. Furthermore, when the user-item interaction is a binary outcome ($r_{u,i} \in \{0, 1\}$), it is natural to use cross-entropy (Equation 3.26) as the loss function for the optimization problem (SALAKHUTDINOV; MNIH, 2007).

$$err_{ui} = r_{u,i} \times \log\left(\widehat{r}_{ui}\right) + (1 - r_{u,i}) \times \log(1 - \widehat{r}_{ui}) \tag{3.26}$$

3.3.3 Incremental Regularized Matrix Factorization (IRMF)

Luo, Xia & Zhu (2012) also focused on the recommender models' incremental ability in their work, proposing a CF method based on the Regularized Matrix Factorization (RMF). The model has two phases; the batch training phase and the incremental update phase. They perform the batch training of the user A_u and item B_i vectors of latent factors using the SGD update rules (see Equation 3.5, respectively). The incremental update phase first simplifies the offline model training to propose the Sequence Independent Regularized Matrix Factorization (SI-RMF). The SI-RMF model is an RMF variant with the removal of sensitivity to the input sequence of training examples and a simple mathematics form for incremental updates. The authors then use the model structure to design two recommender strategies, the Incremental RMF (IRMF) and the Incremental RMF with linear biases (IRMF-B).

3.3.4 Other Incremental Methods

In the work of Yu et al. (2016), instead of use SVD or RMF, the authors developed a novel method for incremental learning of MF models based on Alternating Least Squares (ALS). The proposed method, called One-sided Least Squares, provides accuracy near-equal to ALS at much faster learning speeds. He et al. (2016) also proposed an ALS based recommender model for online matrix factorization updates. The method highlights two critical issues: the

uniform weight on missing data and the offline setting in dynamic scenarios. To solve these issues, they first propose to weigh the missing data based on item popularity and design a new learning algorithm based on the element-wise Alternating Least Squares (eALS) technique. The authors designed an incremental update strategy that instantly updates an MF model given new feedback to enable online recommendations. On the other hand, in Wu, Wang & Cheng (2008), the authors proposed an incremental recommendation algorithm based on Probabilistic Latent Semantic Analysis (PLSA). The method considers the users' long-term interests, reflected from all the interactions that already occurred, and short-term interests that consider the new lately interactions. The PLSA based algorithm also enjoys the flexibility to update users' positive and negative feedback.

Neural networks are also used in incremental learning for recommender systems. Xu et al. (2020) introduce the GraphSAIL framework, a structure to update the Graph Neural Network-based (GNN-based) recommender models incrementally. The framework's purpose is to address the commonly experienced catastrophic forgetting problem that occurs when training a model incrementally, preserving the user's long-term preferences or item's long-term property during incremental model updating. You et al. (2019) also propose using neural network architectures to make dynamic recommendations to users, introducing the Hierarchical Temporal Convolutional Network (HierTCN) method. HierTCN consists of two levels of models. The high-level model uses Recurrent Neural Networks (RNN) to learn and model the users' evolving long-term interests across different sessions. In contrast, the low-level model is implemented with Temporal Convolutional Networks (TCN), utilizing both the long-term interests and the short-term interactions within sessions to predict the next user taste.

3.4 Methods for Handling Concept Drifts

The work of Zhang & Lu (2020) propose a Multi-Trans matrix factorization (MTMF) model with improved time weight to capture temporal dynamics. The author aims to focus on changes in both user's interests and item characteristics over time. MTMF model presents a personalized time weight based on forgetting curve and item similarity and introduces it into the MF model. Next, MTMF models the user and item dynamics by learning the multiple transitions at the

user-factor and factor-item latent space between the continuous-time and past periods. Accordingly, recommendations are generated by a joint objective function solved by a gradient-based alternating optimization algorithm. Similarly, Rabiu et al. (2020) presents a novel Temporal Matrix factorization method that can capture the changes in users' preferences and in item properties that occur over time.

The work of Zheng et al. (2018) introduces a tourism destination recommender system that employs opinion-mining technology to refine user sentiment. It uses temporal dynamics to represents the changes (drifting) over time in user preferences and destination popularity. These elements are then fused with the SVD++ method by combining user sentiment and temporal influence to make accurate user recommendations.

A Hybrid recommender system (CoAWILDA - relying on on adaptive online Latent Dirichlet Allocation (AWILDA)) is proposed in the work of Al-Ghossein et al. (2018), an adaptive collaborative topic modeling approach to model newly available items arriving as a document stream and incremental matrix factorization for CF. The topic model is maintained up to date in an online fashion and is retrained in a batch when a drift is detected using documents automatically selected by an adaptive windowing technique.

In the work of Laghmari, Marsala & Ramdani (2018), the primary purpose is to introduce an adapted Graded Multi-label Classification (GMLC) method to concept-drifting data streams that used to build a temporal recommender system. The authors propose using incremental decision trees in two incremental GMLC models since the GMLC approaches can handle only uni-dimensional data, and collected data for recommender systems are multidimensional (users and items). The first model (H^U), named User-incremental-GMLC is built considering users as instances and user characteristics and item ratings as descriptive attributes. The second model (H^I), named Item-incremental-GMLC is built considering items as instances and item characteristics and user ratings as descriptive attributes.

Concept drifts are also addressed in a Dynamic Recommender System (DRS) proposed by Rana & Jain (2014). The recommendation model considers the users' requirements changing over time in seeking information on the web. The DRS model is based on an evolutionary clustering algorithm that clusters similar users and evolves them to depict accurate and relevant user preferences over time.

Temporal dynamics scenarios, in which recommender models deal with users' preferences for products drifting over time, is the central aspect considered in the work of Koren (2010). The author proposes a more sensitive approach that can distinguish between short and long-term patterns, allowing exploiting the relevant components of all data instances while discarding only whatever is modeled as irrelevant.

Many techniques are equipped with forgetting approaches to make the stream mining algorithms adaptive to changes in recommender systems scenarios. In this sense, Matuszyk et al. (2015b) developed five new forgetting mechanisms for incremental matrix factorization in recommender systems, applied in both rating and item predictions. Additionally, Matuszyk et al. (2018) present unsupervised forgetting techniques that make recommender systems adapt to changes in users' preferences over time. The proposed techniques are subdivided into obsolete information selection and algorithms to enforce forgetting in different ways.

Zhang et al. (2018) provide a dynamic memory-based CF method, a novel two-layer neighbor selection scheme that takes users' capability and dynamic trustworthiness into account on recommendations. The method uses a time factor to evaluate whether a user shares consistent preferences with the target user and also a time window to forget the user's previous ratings.

3.5 Association Rules Methods

Zhou & Hirasawa (2019) propose an online supermarket recommender model using temporal association rules. The method involves implementing metaheuristics, genetic network programming (GNP) to extract the interesting temporal associations from the online customer dataset. Additionally, the model uses the obtained rules to forecast future customer needs and an ant colony optimization (ACO) approach to evolve the online recommender continuously.

3.6 Hybrid Approaches

The Stochastic Gradient Descent (SGD) technique is the most used to updating the latent factors models'. In the work of Lin, Wang & Tsai (2018), the authors present a hybrid real-time incremental stochastic gradient descent (RI-

SGD) updating technique for implicit feedback matrix factorization (MF) recommendation systems. Lin, Wang & Tsai (2018) implement the RI-SGD model on IBM-Streams, a real-time streaming data analytics platform developed by IBM (ZIKOPOULOS; EATON, 2011). The RI-SGD is part of a recommender system composed of four modules: 1) data collector, 2) model trainer, 3) model updater, and 4) real-time recommender. The recommender system performs the model training (module 2) using the ALSWR algorithm for matrix factorization, and then the model updater module uses the RI-SGD approach to the online latent factors update.

Ullah et al. (2012) also proposed a Hybrid Recommender system in their work. Such an approach combines the model-based and memory-based concepts to have better scalability and accuracy, respectively. The recommendation system subdivides the process into offline and online steps. The first step of the temporal-aware hybrid user model offline uses rating similarity, attribute similarity, temporal information, and user demographic information. In the second step, the method recommends online using user similarity to find the neighbor set, temporal information, and rating matrix.

The work of Zhou et al. (2020) proposes a Hybrid Large-scale social recommender model with online updates (LsRec). LsRec systematically integrates matrix factorization technique and online incremental update, leveraging such social network and numerical ratings for the recommendation. Such a hybrid recommender system considers users' social relationships and clusters items according to the similarity measure degree. LsRec model perform prediction and recommendation in each generated item cluster, respectively.

3.7 Final Considerations

This chapter provided an overview of popular batch and streaming recommendation techniques. Most of the presented recommender models are variants of the traditional Matrix Factorization technique, often used in collaborative scenarios. As observed in this chapter, most of the streaming recommendation techniques have as a baseline model some batch approach, which is adapted to deal with dynamic environments. This thesis focuses on the ISGD, IBPRMF, and PMF streaming recommender models as the baselines for the experiments and proposed methods. Additionally, for further analysis, I want to implement some incremental models with drift detectors to adapt their parameters when

explicitly detecting changes in user behavior. Next, Chapter 4 presents the experimental protocol for accessing the recommender models' performance in batch and streaming learning schemes.

4

Experimental Protocol

This chapter describes the experimental protocol used in this thesis to test the provided novel supermarket data collection and analyze the proposed recommendation strategies Section 4.3.

4.1 Datasets

This section introduces the statistics of some datasets used in this work. In Chapter 5, only the provided supermarket data collection (SMDI) (VINISKI et al., 2021) were used. In the optimizers contributions (Chapter 6), were selected four real-word datasets for experimentation: Amazon Books (MCAULEY, 2014), Movie Lens 1M (HARPER; KONSTAN, 2016), Movie Tweetings (DOOMS; PESSEMIER; MARTENS, 2014), and SMDI-200UE (VINISKI et al., 2021). Finally, considering the negative-relevant items sampling strategies (Chapter 7), the Amazon Books, Movie Lens 1M, Movie Tweetings, SMDI-200UE, and TaFeng[1] datasets were used.

The Amazon Books, Movie Lens 1M, and Movie Tweetings datasets present explicit user feedback. To analyze the effectiveness of the proposed models in the One-Class Collaborative Filtering scenario, the explicit feedback were transformed into positive-only feedback. Only those ratings greater than 3.5 were considered positive feedback . Table 4.1 depicts the main characteristics of

[1]https://www.kaggle.com/datasets/chiranjivdas09/ta-feng-grocery-dataset

these datasets, including the number of interactions, users, items, and sparsity, which is given by Equation 4.1, where $|D|$ is the number of unique interactions, m the number of users, and n the number of items.

$$\text{Sparsity (\%)} = 100 \times \frac{1 - |D|}{m \times n} \tag{4.1}$$

Table 4.1: Overview of the datasets used during experimentation.

Datasets	Interactions	Users	Items	Sparsity
Amazon Books	769991	21675	22223	99.84%
Movie Lens 1M	575280	6038	3533	97.30%
Movie Tweetings	658700	65513	28149	99.96%
SMDI-200UE	447391	9472	6924	99.59%
Ta feng	817741	32266	23821	99.90%

4.2 Recommender Models Implementation

We implemented the batch models on top of TensorFlow (ABADI et al., 2016), as most are based on neural network algorithms. On the other hand, the Python libraries were used to implement the incremental (streaming) models. All experiments were performed on an Intel i7-based computer with 64GB of RAM, an NVIDIA Titan V with 12 GB of RAM, and an NVIDIA RTX 2070 SUPER with 8GB of RAM.

Based on the item prediction strategy, the Binary Cross-Entropy (Log-loss Equation 4.2) loss function were used for the recommender models.

$$\text{Log-loss} = -\frac{1}{|T|} \sum_{u,i \in T} \left(R_{u,i} \log \left(\widehat{R}_{ui} \right) + (1 - R_{u,i}) \times \left(\log(1 - \widehat{R}_{ui} \right) \right) \tag{4.2}$$

where $|T|$ is the number of training or validating samples. For the rating prediction models, the Mean Squared Error (MSE) were used as the loss function, which is depicted in Equation 4.3.

$$\text{MSE} = \frac{1}{|T|} \sum_{u,i \in T} \left(R_{u,i} - \widehat{R}_{ui} \right)^2 \tag{4.3}$$

The model parameters were randomly set according to a Gaussian distribution with $\mu = 0$ and a $\rho = 0.01$. Hyper-parameter tuning was performed on a recommender model and dataset basis. As this thesis presents three contributions (Chapters 5, 6, and 7), the datasets, recommender models, and the set of parameters used in each contribution changes. The analysis explored different recommendation challenges, which require a set of specific parameters depending on the analyzed scenario. However, the following sections present the protocol used for dataset split and recommender techniques evaluation.

4.3 Proposed Protocol for Batch and Streaming Comparison

This section describes the experimental protocol used to compare batch and streaming algorithms in the SMDI dataset (Chapter 5). This experimental protocol is relevant to guarantee that batch and streaming methods are adequately compared and enable identifying concept drifts and cold start problems.

Figure 4.1 shows the proposed batch and stream protocols. The dataset was split using the first two months of data for training and the remainder two months for testing. The temporal split makes more sense than a random one because users' interests may change over time (MATUSZYK et al., 2015a). It is also more realistic as it mimics the data behavior if any recommender systems were applied in the real-world (SIDANA et al., 2017), thus comparing batch and stream learning algorithms fairly.

During the training step, batch and streaming algorithms have significant differences. The batch training first shuffles the data and use 20% of it for validation. The validation set is applied to monitor the validation loss, thus allowing training early stopping. Finally, the test set were used for assessing the recommender system on unseen data.

Regarding streaming models, the first 20% of the training set is used solely for training. The rationale is to allow the streaming recommender system to learn initial parameters and uncover user-item relationships embedded within the latent factors and non-random output recommendations at the beginning of the experiment. We use the remainder of the training set for testing and incremental training. Data shuffling is not performed as the instances' natural order must be preserved (VINAGRE; JORGE; GAMA, 2014). Next, the test set is

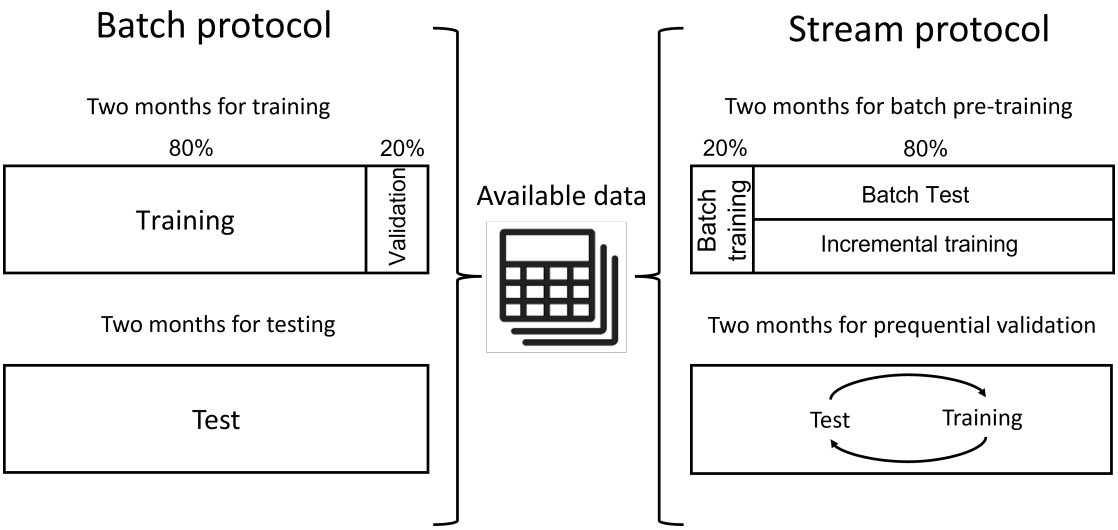


Figure 4.1: Batch and stream protocols.

used in a test-then-train fashion, meaning that user-item interaction is queried and later used for model update (GAMA; SEBASTIÃO; RODRIGUES, 2013).

4.4 Protocol Used for the Proposed Recommendation Strategies

Our focus in the proposed recommendation strategies (Chapters 6 and 7) is the collaborative filtering approach in incremental (streaming) fashion. Because the main focus of this thesis is a streaming scenario in which the order of the events is significant, the datasets were splitted according to the protocol suggested in (GAMA; SEBASTIÃO; RODRIGUES, 2013).

The first 20% of each dataset were used for batch training, 30% for batch testing and incremental training, and the remaining 50% as the test set. The test set is used in a test-then-train fashion, meaning that each user-item interaction is used sequentially for model evaluation and updating. Figure 4.2 presents the protocol for assessing the recommender models' accuracy used to analyze the proposed recommendation strategies.

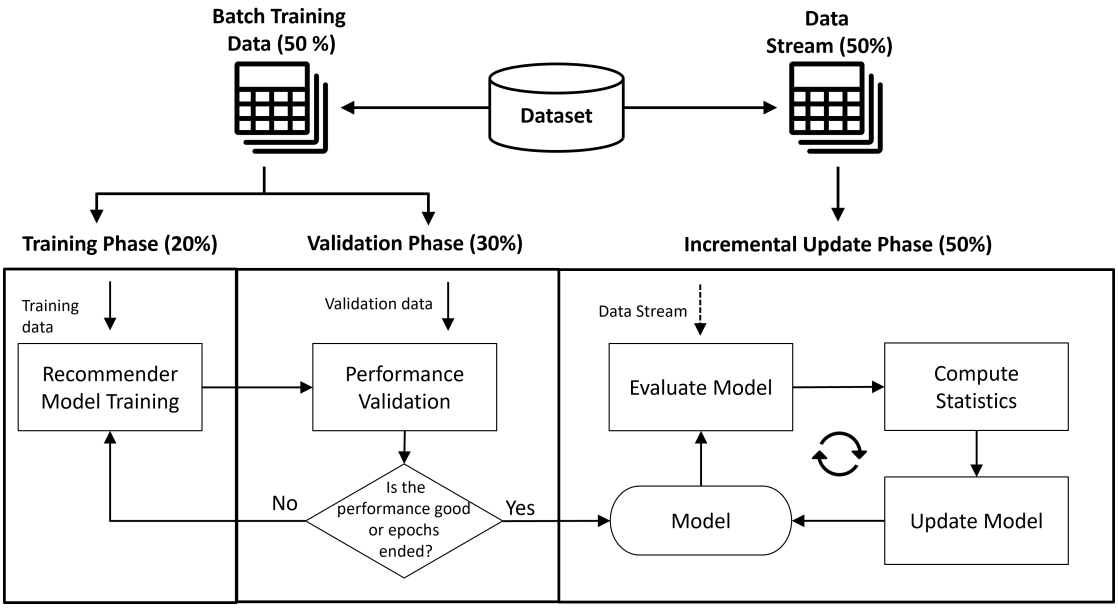


Figure 4.2: Stream protocol.

4.5 Evaluation

This section introduces the metrics and the evaluation variants used in our project to test the recommender methods.

4.5.1 Evaluation Metrics

We express the goodness-of-fit of the models using `RECALL@K` (CREMONESI; KOREN; TURRIN, 2010a) and normalized discounted cumulative gain (`NDCG@K`) (HE et al., 2015) metrics with K $\in \{10\}$, which are the most commonly used thresholds. The `RECALL@K` metric quantifies how often a recommender suggests a relevant item (hit) amongst unknown items, which are assumed to be irrelevant.

For each instance (u, i) in the test set (T), were selected a candidate list of 1000 unknown items for user u, and the known (relevant) item i was appended to this candidate list. We ordered candidate items by descending proximity to a value of 1 (as the focus is the positive-only scenario) using the function $f_{ui} = |1 - \widehat{R}_{ui}|$, according to the non-Boolean predicted scores $\widehat{R}_{ui}$ obtained by the recommender models.

The `RECALL@K` metric, described in Equation 4.4, measures the average (across all users) of the proportion of recommended items that appear among the top K positions of the ranked list (YUAN; CHEN; ZHAO, 2011), where $|T|$ is the test

set size.

$$\text{RECALL@K} = \frac{1}{|T|} \sum_{(u,i) \in T} \text{hit@K}(u, i) \tag{4.4}$$

For each instance $\langle u, i \rangle$, hit@K$(u, i) = 1$ is said to happen when i is ranked among the top K items, and hit@K$(u, i) = 0$ otherwise.

The `NDCG@K` metric, described in Equation 4.5, indicates whether the ground-truth items are ranked higher than others by accounting for the position of hit (HE et al., 2015).

$$\text{NDCG@K} = \begin{cases} 0, & \text{if rank(i)} \leq \text{K} \\ \frac{1}{log_2(rank+1)}, & \text{otherwise} \end{cases} \tag{4.5}$$

4.5.2 Evaluation Variants

The protocol followed to assess RECALL, and NDCG has two variants, presented in Figure 4.3. The first is an approach referred as a 'basic evaluator,' which measures the recommender system using the entire test set. This approach allows the comparison between recommender systems and hypothesis testing. The second approach is the 'window-based evaluator,' which reports the recall over test set chunks. We use a window with size at every 1% of the test set (Figure 4.3).

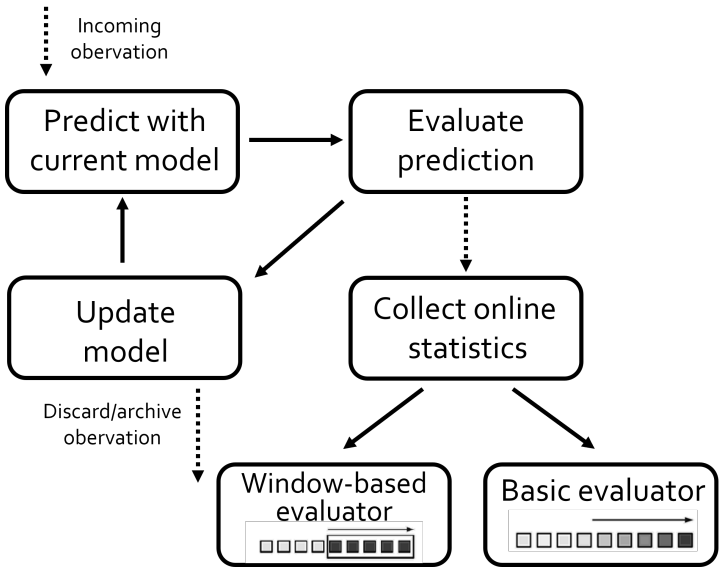


Figure 4.3: Prequential Validation. Adapted from (JORGE et al., 2016).

The rationale behind the window-based evaluation is that it allows the assessment of recommender systems over time. This assessment is critical to verify whether the dataset exhibits drifting characteristics and whether streaming models benefit from the incremental updates performed over test data. Therefore, the Prequential test-then-train process (GAMA; SEBASTIÃO; RODRIGUES, 2013; JORGE et al., 2016) were followed for validating streaming models as depicted in Figure 4.3.

4.5.3 Significance Test

Finally, the hypothesis testing were incorporated to determine whether one recommender algorithm outperforms others. This thesis experiments followed the protocol reported in (DEMSAR, 2006) by combining Friedman (FRIEDMAN, 1937) and the Nemenyi post-hoc (NEMENYI, 1963) statistical tests. The experimental results are the mean and standard deviation of the experiments' replications. To analyze the significance of test results, in Chapters 6 and 7, were also plotted the critical distance (CD) of the experiment results, which helps to determine if there are significant differences between the analyzed methods.

4.6 Final Considerations

This chapter presented the experimental protocol for accessing the recommender models' performance in batch and streaming learning schemes. The introduced protocol were used in all the contributions presented in the following sections. Next, Chapter 4 presents the collected, processed, and provided supermarket data collection.

5

Contribution I - Supermarket Data Collection

In this chapter, we focus on two problems that affect recommender systems. The first is *concept drift*, which refers to changes in the data behavior over time (TSYMBAL, 2004; WEBB et al., 2018). In recommender systems, concept drift reflects changes in the interactions between customers and items, either because (i) customers' preferences change, (ii) new items become available for purchase, etc. The second is *cold start*, which occurs when new customers or items appear in the recommendation scenario. Such a problem is challenging, because the recommender model cannot make robust inferences for users or items about which it has not yet collected enough information (OCEPEK; RUGELJ; BOSNIC, 2015; SHAO; LI; BIAN, 2021).

Recommender systems are traditionally trained in a batch fashion, which means that given a training set composed of interactions between users and items, a static model is learned and deployed *ad eternum*. Consequently, it is relevant to tailor recommender systems that can be incremented over time, assuming that the interactions between users and items are made available as a stream of events.

In this chapter, our goal is to bring forward a case study of existing recommender algorithms in a real-world supermarket scenario that exhibits both concept drifts and cold start problems. The contribution is threefold, as follows:

- A comparison of existing filtering recommender systems in which the impact of concept drift and cold start is assessed in both batch and streaming

fashions.

- Evidence that the incremental ability of the streaming-based recommender systems allows a better recovery when cold start is present.
- A novel real-word supermarket dataset that exhibits concept drifts and cold start problems is made publicly available.

This chapter is divided as follows. Section 5.1 details a new dataset we make available regarding supermarket transactions, which exhibits concept drift and cold start characteristics. Section 5.2 describes the experiments undertaken to perform the proposed analysis of recommender systems. Section 5.3 analyzes existing works in recommender systems to answer whether streaming recommender systems overcome batch approaches w.r.t. concept drifts and cold start. Finally, Section 5.4 concludes this chapter.

5.1 SDMI Dataset

This section describes the Supermarket Dataset with Implicit Feedback (SMDI) used in this case study, broadening data acquisition, pre-processing, and descriptive statistics.

5.1.1 Data Acquisition

The original dataset came from a supermarket that is a customer of HiMarket[1], and it encompasses purchases made between August 1st, 2019, and November 30th, 2019. It contains 737,893 events representing 9,531 customers purchasing 7,151 items in supermarket transactions carried out on-site since the customer analyzed does not provide delivery services. The events are sorted chronologically and reported in the ⟨user; item; rating; timestamp⟩ form, as illustrated in Figure 5.1.

Figure 5.2 provides insights on purchases' temporal traits considering different granularities (day, week, and month). In the daily plot provided in Figure 5.2a, we observe that the number of items sold over time drastically reduces. The reason is that the beginning of the data timespan matches the rewards club launch and that the interest decreased over time. Furthermore, we observe that

[1]Himarket is a company located at Curitiba, Paraná, Brazil. HiMarket's website is available at <http://himarket.club/>

Figure 5.1: Dataset representation.

user_id	item_id	rating	timestamp
1078	4175	1	1564628400
2317	6535	1	1564628400
2317	1499	1	1564628400
2317	4175	1	1564628400
2317	123	1	1564628400
...	...	...	...
4901	687	1	1575082800
4901	496	1	1575082800
1026	2827	1	1575082800
1026	6960	1	1575082800
4198	2531	1	1575082800

the number of purchases increases during weekends (Figure 5.2b) and in the first and last three days of the month (Figure 5.2c[2]).

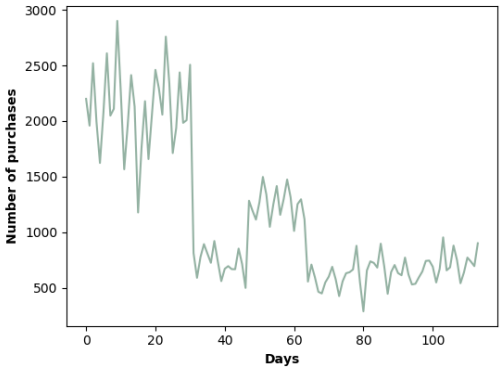
One particular consideration of the SMDI dataset is that it contains repeated events, i.e., users may purchase the same item multiple times. Consequently, Figure 5.3a depicts the dataset 'imbalance' as the maximum and minimum amount of user interactions is volatile. On the other hand, Figure 5.3b shows that some items have a higher number of interactions than others, as these are essential items. Given the characteristics mentioned above, there are two problems related to the supermarket process that contribute to its 'imbalancedness.' First, we observe that cashiers enter their identification number to enable discounts to customers who do not provide their rewards club code when making purchases. Consequently, some users possess an unrealistic number of purchases. Second, users with a large number of events represent merchants in the region close to the supermarket. Thus, the next section presents two approaches to address these problems and reduce the dataset's imbalance characteristic.

5.1.2 Dataset Pre-processing Approaches

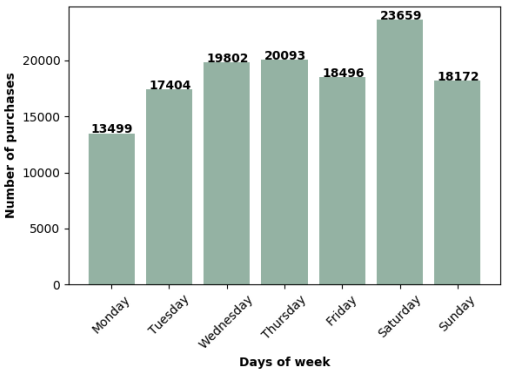
This section presents two pre-processing approaches to alleviate the 'imbalancedness' in the number of user events in the SMDI dataset. These strategies account for the total number of events per user and the number of unique events

[2]An exception regards the 31st day, as not all months have 31 days.

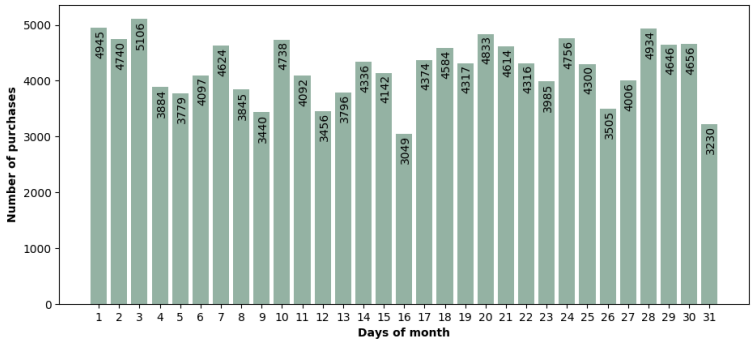
Figure 5.2: Number of purchases considering different timestamp granularity.



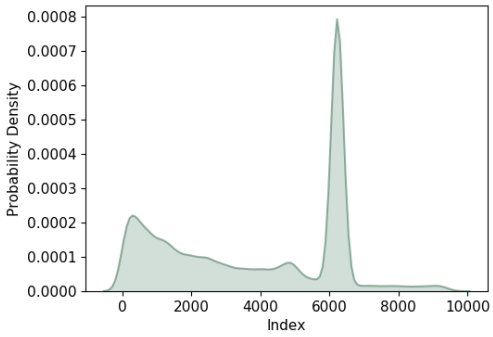
(a) Daily number of purchases.



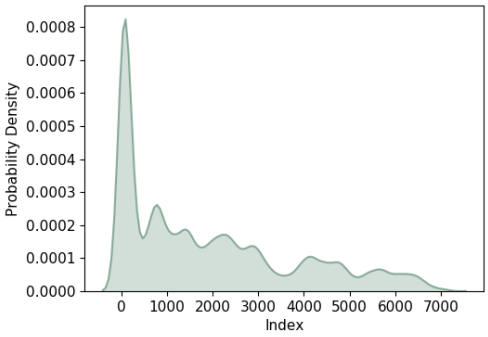
(b) Number of purchases per day of week



(c) Number of purchases per day of month



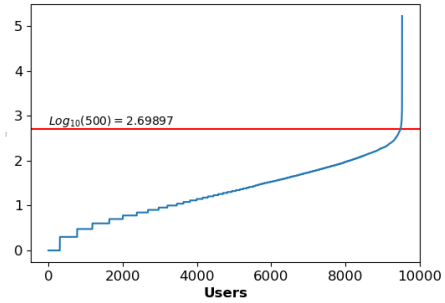
(a) Number of user interactions.



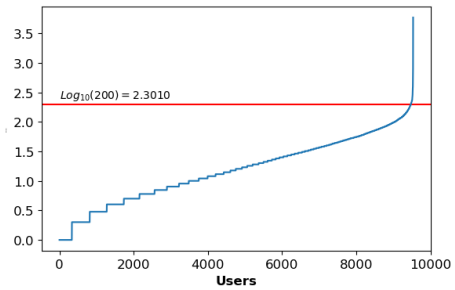
(b) Number of items occurrences.

Figure 5.3: Probability density for users (a) and items (b) in the original SMDI dataset.

per user, respectively.



(a) Distribution of the number of events (N) per user.



(b) Distribution of the number of unique events per user.

Figure 5.4: $\log_{10}$ transformation applied to ordered events (N) for each user (a) and unique events per user (b).

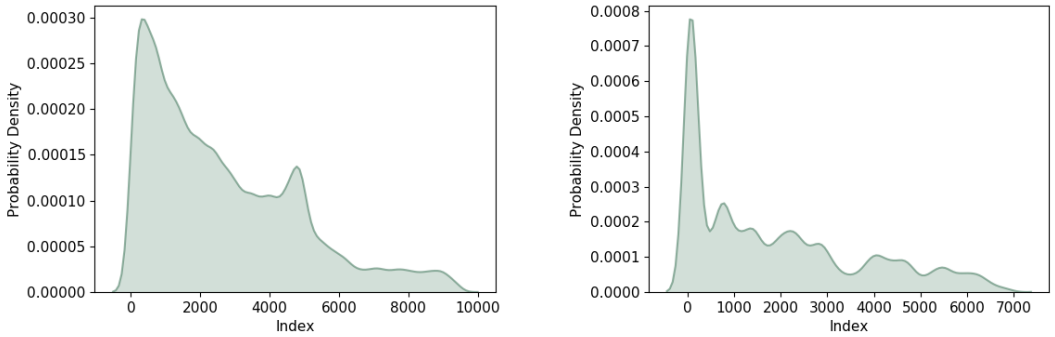
Cut-off point by the total number of events per user

The first approach filters the dataset according to the maximum number of events per user. Figure 5.4a reports the $\log_{10}$ of the number of events per user. In this strategy, we assume as a cut-off point the region of the curve in which the number of interactions substantially rises, i.e., where the approximate value of $\log_{10}$ is 2.7; thus indicating a maximum number of approximately 500 ($\log_{10}(500) = 2.69897$) interactions per user. We refer to this version of the dataset as `SMDI-500E`.

The resulting density distributions obtained for users and items with the removal of such users are depicted in Figures 5.5a and 5.5b, respectively. Despite the removal of users in this pre-processing approach, the distribution observed for the number of items (Figure 5.5b) does not largely differ from the one observed in the original dataset (Figure 5.3b). On the other hand, with the removal of users with more than 500 interactions, the 'imbalancedness' was reduced. In Table 5.2, we observe the characteristics of the pre-processed dataset. We observe that the maximum number of interactions per user is 500, regardless of whether there are multiple interactions with the same items.

Cut-off point by the number of unique events per user

When analyzing the number of unique interactions per user in the original dataset in Figure 5.4b and Table 5.2, we also observe another relevant imbalance



(a) Probability distribution of users' occurrences.

(b) Probability distribution of items' occurrences.

Figure 5.5: Probability distribution for users (a) and items (b) in the `SMDI-500E` dataset.

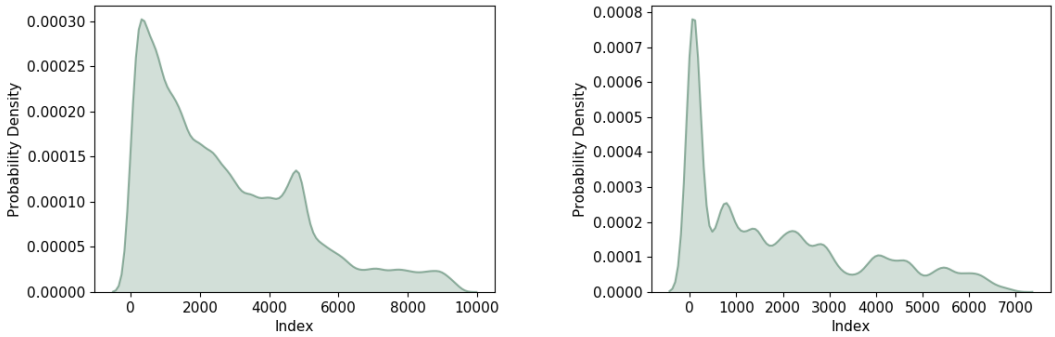
in the dataset. More specifically, the number of unique events for cashiers is close to the total number of items available in the dataset and reinforces the need to remove such users.

Following the same rationale for picking a cut-off point, an analysis of Figure 5.4b showed that the number of unique events per user rapidly increases with an approximate value of 2.3 for the $\log_{10}$ transformation of the number of unique events, thus indicating the removal of users who interacted with more than 200 ($\log_{10}(200) = 2.3010$) different items in the analyzed period. Consequently, we refer to this dataset as `SMDI-200E` in the remainder of the chapter.

The probability distribution for users and items obtained after this pre-processing approach are given in Figures 5.6a and 5.6b, respectively. Similarly to the previous section's approach, a significant reduction in the number of interactions per user has been observed (Figure 5.6a). At the same time, the probability curve for items (Figure 5.6b) does not significantly differ from the one observed in the original dataset (Figure 5.3b). Table 5.2 depicts this variant characteristics.

5.1.3 Dataset Availability and Content

Table 5.1 overviews the dataset's content, particularly the included files, their respective formats, and pieces of information. The dataset, its pre-processed versions, and the source code to replicate the experiments described in this chapter can be downloaded from <https://dcam.ppgia.pucpr.br/assets/datasets/



(a) Number of user interactions. (b) Number of items occurrences.

Figure 5.6: Probability density for users (a) and items (b) in the `SMDI-200E` dataset.

SMDI700kDataset.zip>. The original events without any pre-processing are in the `SMDI_original.csv` file. The `SMDI-500E.csv` and `SMDI-200UE.csv` files contain the results of pre-processing approaches shown in Section 5.1.2 and 5.1.2, respectively. Users and items information are presented in `SMDI_users.csv` and `SMDI_items.csv`, respectively.

Table 5.1: Description of the files constituting the SMDI datasets.

File	Format	Content
`SMDI_original`	csv	*user_id*,**item_id**, rating, timestamp
`SMDI-500E`	csv	
`SMDI-200UE`	csv	
`SMDI_users`	csv	*user_id*, R, F, M, RFM, R1, F1, M1, RFM1, R2, F2, M2, RFM2, R3, F3, M3, RFM3, R4, F4, M4, RFM4
`SMDI_items`	csv	**item_id**, section_id, brand_id, ref_price, avg_price, min_price, max_price, amount

On the item information level, the SMDI dataset includes section and brand identifiers (`section_id` and `brand_id`), reference price (`ref_price`), average, minimum and maximum prices during the period (`avg_price`, `min_price`, and `max_price`), and amount. Since users' personal information is not made available and only purchase data was collected, we calculate, based on the obtained data, the Recency-Frequency-Monetary (`RFM`) score (WENG, 2017). Recency (`R`) is the interval between the current and previous purchase, and thus, the shorter the interval is, the bigger `R` is. Frequency (`F`) indicates the user's number of transactions in a particular period. The bigger the frequency is, the bigger `F`

is. Finally, Monetary (`M`) refers to all transactions' monetary value in a specific period.

The RFM analysis assigns three different scores related to recency, frequency, and monetary variables to each customer, using a scale from 1 to 5. Therefore, the database is sorted per `RFM` dimension, and the customer list is divided into five equal segments. The top quintile is assigned a score of 5, and the others receive 4, 3, 2, and 1 (CHRISTY et al., 2018), according to the quintile they belong. The `RFM` score is then generated by concatenating `R`, `F`, and `M` components, in this specific order. The normalized recency, frequency, and monetary values, calculated monthly (variables R1, F1, M1; R2, F2, M2; and so forth) and the entire period's value are available in the `SMDI_users.csv` file.

5.1.4 Descriptive Statistics

Table 5.2 provides statistics from the variants of the SMDI dataset. Regarding the pre-processing methods discussed in Sections 5.1.2 and 5.1.2, we see that both pre-processing approaches obtained close values to the total number of interactions (448791 and 447391), as well as the number of users (9480 and 9472) and items (6933 and 6924). Conversely, the maximum number of events per user in the `SMDI-500E` dataset is 775, which is much higher than 491 seen in the `SMDI-200UE` variant. For both pre-processed datasets, the average number of interactions (Avg # of events) per user was the same (47).

Table 5.2 also presents the statistics for each dataset w.r.t. unique events. Despite the maximum number of unique events per user in the original dataset (5853) being much higher than for pre-processed datasets (270 for both), the average number of unique events per user (Avg # of events) for both is roughly the same (30 for the SMDI_original dataset and 28 for the `SMDI-500E` and `SMDI-200UE` datasets).

5.2 Experimental Setup

This section describes the experimental protocol used to compare batch and streaming algorithms in the SMDI dataset. For the NCF models (GMF, MLP, and NeuMF), instead of only using the positive examples to modeling the relationship between users and items, we randomly sampled four unknown items per positive example to serve as negative ones according to the protocol suggested

Table 5.2: Description of the SMDI dataset variants.

Information	**original**	**SMDI-500E**	**SMDI-200UE**
Events	737,893	448,791	447,391
Unique events	292,943 (39.7%)	268,592 (59.84%)	266,354 (59.84%)
Users	9,531	9,480	9,472
Items	7,141	6,933	6,924
Avg # of events	77	47	47
Min # of events	1	1	1
Max # of events	166,549	491	775
Sparsity	99.57%	99.59%	99.59%
Avg # of unique events	30	28	28
Min # of unique events	1	1	1
Max # of unique events	5,853	270	200

in (HE et al., 2017). Considering the paired model BPRMF, it requires a single negative instance per positive interaction during training. Thus, we randomly selected a negative example to balance the positive-negative item pairs.

We tested the following hyper-parameter values for SVD, BPRMF, ISGD, and IBPRMF: learning rate (learning-rate) $\in$ [0.01, 0.02, 0.05, 0.001, 0.005, 0.0001, 0.0005], regularization rate (reg-rate) $\in$ [0.01, 0.001, 0], latent factors $\in$ [5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100] for SVD, BPRMF, ISGD and IBPRMF. For GMF, MLP, and NeuMF models, we tested the values of latent factors $\in$ [8, 16, 32, 64, 128], and as suggested in the original paper (HE et al., 2017), the embedding layers do not have regularization, i.e., reg-rate= 0. The number of negative items selected for the NCF models (GMF, MLP and NeuMF). In Table 5.3, we show the best parameters selected per model and dataset.

We evaluate our proposed dataset variants using the RECALL@N metric (Equation 4.4). The main difference from the protocol presented in Chapter 4 is the size of candidate list, here we selected a candidate list of 100 unknown items to user u. We also use both the 'basic evaluator,' and 'window-based evaluator,' evaluation strategies presented in Chapter 4. The experimental results are the mean and standard deviation of 30 replications.

5.3 Experimental Results and Analysis

This section reports the experimental results observed when comparing batch and streaming recommender algorithms applied to the SMDI datasets. We dis-

Table 5.3: Parameters tuning for each model and dataset.

Dataset	Model	Opt	Loss	factors	reg-rate	learning-rate
SMDI_original	SVD	SGD	MSE	40	0.01	0.001
	BPRMF	SGD	MSE	30	0.01	0.0005
	GMF	Adam	Log-loss	32	0	0.001
	MLP	Adam	Log-loss	32	0	0.0001
	NeuMF	Adam	Log-loss	32	0	0.0005
	ISGD	SGD	MSE	10	0.01	0.02
	IBPRMF	SGD	MSE	20	0.001	0.001
SMDI-500E	SVD	SGD	MSE	30	0.01	0.001
	BPRMF	SGD	MSE	40	0.01	0.0001
	GMF	Adam	Log-loss	32	0	0.001
	MLP	Adam	Log-loss	32	0	0.001
	NeuMF	Adam	Log-loss	32	0	0.001
	ISGD	SGD	MSE	10	0.01	0.02
	IBPRMF	SGD	MSE	30	0.001	0.05
SMDI-200UE	SVD	SGD	MSE	40	0.01	0.001
	BPRMF	SGD	MSE	80	0.01	0.0005
	GMF	Adam	Log-loss	32	0	0.001
	MLP	Adam	Log-loss	32	0	0.0001
	NeuMF	Adam	Log-loss	32	0	0.0001
	ISGD	SGD	MSE	10	0.01	0.02
	IBPRMF	SGD	MSE	40	0.001	0.05

cuss the observations of the two proposed strategies planned in the experimental protocol, as follows: the basic evaluation in Section 5.3.1 and the window-based evaluation in Section 5.3.2. These results are related to our **Hypothesis #1.** *Adaptive and incremental methods present superior performance compared with batch processing techniques in datasets with concept drift and high incidence of cold-start.* Therefore, we want to show that the streaming recommender models, ISGD and IBPRMF, perform better than the batch models in datasets with the incidence of cold-start and concept drift.

5.3.1 Results of the Basic Evaluation

This section reports the results obtained by batch and streaming models in the SMDI dataset variants. More specifically, we focus on the basic evaluation process described in the proposed experimental protocol (Section 5.2) in which the recommendation rates are computed over the entire test set. Table 5.4 shows the general performance obtained.

According to the dataset variants' results, we observe differences in the tested recommender models' behavior. Considering the original dataset results, we verify that there is no clear indication of whether batch or streaming models outperform others. We report the statistical test results obtained by combining Friedman and Nemenyi tests assuming a 95% confidence level throughout our discussion. These differences are also reported in Table 5.4, where recall values are marked with letters ($a \succ b \succ c \succ d \succ e \succ f$) that depict and group the goodness of their results.

Table 5.4: Recall values obtained by the recommendation methods in each tested dataset. The shaded area comprises the results of the data stream algorithms.

Model	RECALL@1	RECALL@5	RECALL@10	RECALL@20
SMDI_original.csv				
PopTop	0.041_{d}	0.102_{d}	0.150_{e}	0.215_{f}
SVD	$\mathbf{0.325 \pm 0.0021}_{a}$	$\mathbf{0.565 \pm 0.0012}_{a}$	$0.669 \pm 0.0011_{b}$	$0.779 \pm 0.0009_{b,c}$
BPRMF	$0.242 \pm 0.0013_{b,c}$	$0.410 \pm 0.0018_{c}$	$0.484 \pm 0.0019_{d}$	$0.564 \pm 0.0014_{d,e}$
GMF	$0.218 \pm 0.0023_{c}$	$0.376 \pm 0.0037_{c}$	$0.454 \pm 0.0048_{d}$	$0.542 \pm 0.0055_{e}$
MLP	$\mathbf{0.324 \pm 0.0032}_{a}$	$0.559 \pm 0.0086_{b}$	$0.665 \pm 0.0096_{b,c}$	$\mathbf{0.790 \pm 0.0116}_{a,b}$
NeuMF	$\mathbf{0.322 \pm 0.0125}_{a}$	$0.559 \pm 0.0243_{a,b}$	$0.664 \pm 0.0257_{b,c}$	$\mathbf{0.790 \pm 0.0268}_{a}$
ISGD	$0.298 \pm 0.0023_{b}$	$0.562 \pm 0.0012_{b}$	$\mathbf{0.674 \pm 0.0027}_{a}$	$\mathbf{0.782 \pm 0.0020}_{a,b}$
IBPRMF	$\mathbf{0.324 \pm 0.0144}_{a}$	$0.559 \pm 0.0075_{b}$	$0.662 \pm 0.0046_{c,d}$	$0.766 \pm 0.0038_{c,d}$
SMDI-500E.csv				
PopTop	0.042_{f}	0.102_{f}	0.150_{f}	0.216_{e}
SVD	$0.299 \pm 0.0019_{c}$	$0.543 \pm 0.0010_{b,c}$	$0.648 \pm 0.0011_{c}$	$0.757 \pm 0.0015_{c,d}$
BPRMF	$0.212 \pm 0.0031_{d,e}$	$0.382 \pm 0.0046_{d,e}$	$0.457 \pm 0.0050_{d,e}$	$0.538 \pm 0.0057_{d}$
GMF	$0.197 \pm 0.0025_{e}$	$0.371 \pm 0.0038_{e}$	$0.454 \pm 0.0041_{e}$	$0.540 \pm 0.0046_{d}$
MLP	$0.299 \pm 0.0108_{c,d}$	$0.539 \pm 0.0020_{c,d}$	$0.645 \pm 0.0016_{c,d}$	$\mathbf{0.779 \pm 0.0019}_{a,b}$
NeuMF	$0.293 \pm 0.0220_{b,c}$	$0.543 \pm 0.0057_{b,c}$	$0.650 \pm 0.0027_{b,c}$	$\mathbf{0.782 \pm 0.0030}_{a}$
ISGD	$\mathbf{0.317 \pm 0.0008}_{a,b}$	$\mathbf{0.571 \pm 0.0004}_{a}$	$\mathbf{0.676 \pm 0.0004}_{a}$	$\mathbf{0.782 \pm 0.0005}_{a}$
IBPRMF	$\mathbf{0.322 \pm 0.0007}_{a}$	$\mathbf{0.565 \pm 0.0006}_{a,b}$	$\mathbf{0.667 \pm 0.0006}_{a,b}$	$0.772 \pm 0.0006_{b,c}$
SMDI-200UE.csv				
PopTop	0.042_{e}	0.102_{e}	0.150_{e}	0.216_{e}
SVD	$0.299 \pm 0.0023_{b}$	$0.544 \pm 0.0012_{b,c}$	$0.648 \pm 0.0011_{c}$	$0.757 \pm 0.0016_{c,d}$
BPRMF	$0.213 \pm 0.0028_{c}$	$0.384 \pm 0.0029_{c,d}$	$0.459 \pm 0.0031_{c,d}$	$0.539 \pm 0.0040_{d}$
GMF	$0.198 \pm 0.0026_{c}$	$0.371 \pm 0.0031_{d}$	$0.453 \pm 0.0036_{d}$	$0.539 \pm 0.0044_{d}$
MLP	$0.285 \pm 0.0586_{b}$	$0.544 \pm 0.0072_{b}$	$0.666 \pm 0.0029_{b}$	$\mathbf{0.785 \pm 0.0013}_{a}$
NeuMF	$0.292 \pm 0.0235_{b}$	$0.542 \pm 0.0140_{b}$	$0.667 \pm 0.0035_{b}$	$\mathbf{0.784 \pm 0.0011}_{a}$
ISGD	$\mathbf{0.316 \pm 0.0009}_{a}$	$\mathbf{0.570 \pm 0.0005}_{a}$	$\mathbf{0.676 \pm 0.0005}_{a}$	$\mathbf{0.782 \pm 0.0005}_{a,b}$
IBPRMF	$\mathbf{0.322 \pm 0.0006}_{a}$	$\mathbf{0.565 \pm 0.0008}_{a}$	$0.667 \pm 0.0007_{b}$	$0.772 \pm 0.0007_{b,c}$

In this table, we also report the results for PopTop (CREMONESI; KOREN; TURRIN, 2010a), a non-machine learning approach for assessing recommender systems. PopTop consists of recommending items with the best degree of success among all users instead of modeling the user-item relationship using a machine learning method. Consequently, we observe that PopTop is outperformed by all methods regardless of the recall metric analyzed. This observation depicts that both batch and streaming algorithms model the users' behavior in supermarket purchases adequately and surpass a naive baseline.

When comparing the batch models, we observe that two neural network

models, i.e., MLP and NeuMF, outperform GMF, BPFMF, and SVD in most datasets and recall values. These results indicate that the neural networks are suitable to capture complex interactions between users and items. In all datasets, MLP and NeuMF models do not depict a significant difference between each other. Even though NeuMF is a combination of MLP and GMF, GMF alone did not result in better performance as the recall rates observed are not significantly higher than those surveyed for MLP alone.

Regarding the streaming recommender models' results (highlighted in Table 5.4), ISGD outperformed their batch version (SVD) concerning the `RECALL@10` values in all datasets, thus showing a statistically significant difference with a 95% confidence level. The improvement is more significant in the pre-processed datasets than in the original one. We observe an increase of 2.8 percentage points for `RECALL@10` in both `SMDI-500E` and `SMDI-200UE`, in contrast to 0.5 percentage points in the original dataset. The discrepancy between streaming algorithms and the corresponding batch counterpart depicts the importance of constantly updating the recommender system as new data becomes available. This claim is further backed up as ISGD obtained superior results when compared to MLP and NeuMF in the pre-processed variants considering `RECALL@1`, `RECALL@5`, and `RECALL@10`.

It is also noteworthy the analysis between BPR batch and streaming variants, especially in the pre-processed datasets. For instance, IBPRMF increased the `RECALL@10` values up to 21 percentage points compared to the batch model BPRMF, while ISGD improved SVD rates by 2.8 percentage points. Extending the analysis, the `RECALL@10` and `RECALL@20` results obtained in the original dataset show that IBPRMF had performance decreases. These decreases were due to the volatility of the recall rates observed, as IBPRMF did not converge in all experiment runs.

Overall, we observe that streaming models performed competitive results in all the datasets, except the ISGD model in the original dataset when `RECALL@1` is assessed. In this specific scenario, we observe that IBPRMF is a formidable contender to match traditional matrix factorization techniques (SVD) and even more complex approaches that rely on neural networks (MLP and NeuMF). Summing up, we see that in smaller N values, i.e., $N \in [1, 5, 10]$, either ISGD or IBPRMF overcome batch models in the `SMDI-500E` and `SMDI-200UE` variants and that MLP and NeuMF achieve superior results in `RECALL@20`. Comparing the results acquired for both recommendation approaches, i.e., batch and streaming,

we observe that streaming models resulted in improvements of 2.8 percentage points in recall values. In the next section, we further analyze these results from a different perspective. More specifically, we focus on the performance assessment that takes place over time, thus allowing a more fine-grained analysis on why streaming recommender systems exhibit the behavior mentioned above and provide evidence of the existence of concept drifts and cold start in the dataset assessed.

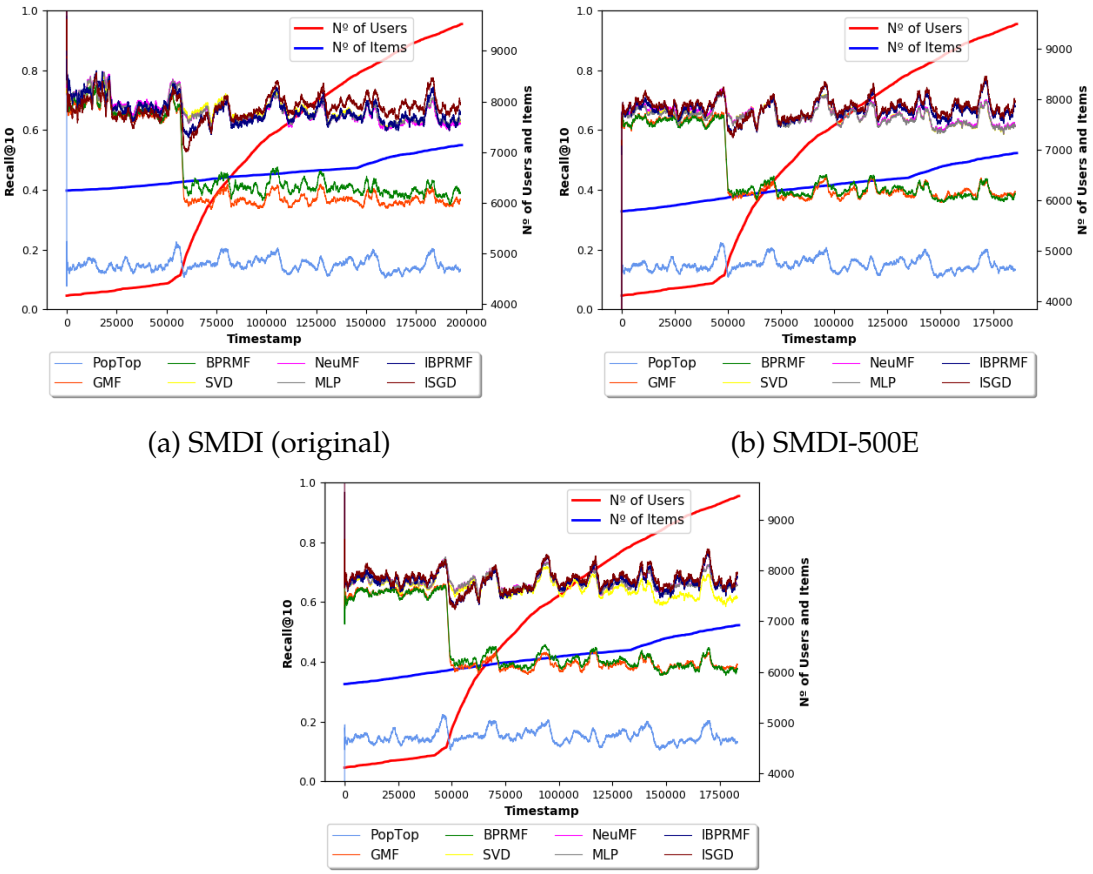
5.3.2 Results of the Window-based Evaluation

In this section, we report the recall rates using the window-based process. Figure 5.7 shows the results obtained with the assessment taking place at every 1% of the test set. In the same figure, we also report the cumulative number of users and items to verify the cold start problem's impact.

In real-world datasets, the assessment of user-item interactions over time may uncover concept drifts, such as the launch of a product that reduces the popularity of previous versions of the same product or its competitors. Consequently, users' interests and preferences may drift over time, resulting in concept drifts that should be targeted by adaptive recommender systems (MATUSZYK et al., 2015a; CHANG et al., 2017). In Figure 5.7, we observe recurrent fluctuations in the `RECALL@10` rates, which represent concept drifts. This observation is corroborated by the recall rates obtained by PopTop, thus depicting that as new user-item interactions occur, the overall behavior in the dataset also changes. Such changes are weekly recurring drifts (GAMA et al., 2014), thus meaning that the relationship between users and items changes over the week, but it repeats itself across weeks. Recurring concepts are expected in supermarket scenarios as specific sales are repeated along weeks, days of the month, or even months of the year.

In this analysis, we observe that the streaming recommender models, i.e., ISGD and IBPRMF, allowed significant parameter adjustments over time that induced better performance when compared to other models, especially in the pre-processed dataset variants. Another relevant aspect observed in Figure 5.7 regards the performance decrease observed after the processing of 50 thousand interactions. This decrease matches the behavior change of the cumulative number of users in the dataset, thus culminating in a cold start problem.

However, even though we notice an abrupt increase in the number of users



(a) SMDI (original)

(b) SMDI-500E

(c) SMDI-200UE

Figure 5.7: Moving averages of `RECALL@10` values in the test stage, when using a sliding window with size 2000; a) shows the plot evolution obtained in the original dataset; b) pre-processed SMDI-500E dataset; and c) pre-processed SMDI-200UE dataset.

in all datasets, most algorithms recover from the cold start and maintain good performance as new instances appear, except for the BPRMF and GMF methods. These results show that (i) streaming models, despite built on matrix factorization, recover from cold start issues swiftly and that (ii) recommender models based on neural networks exhibit interesting behavior in cold-start scenarios even though they are not continuously updated. The reason behind this behavior is related to the internal learning process of neural network-based recommender systems, where the user-item interactions result in higher-order embeddings that better generalize the underlying behavior between users and items when compared to traditional matrix factorization. Consequently, given the neural networks' generalization ability, the recommender models extract

unseen patterns in user-item interactions and provide useful suggestions in cold-start scenarios.

On the other hand, when we analyze the cumulative number of items, we observe that the increase is gradual. Comparing the behavior between the original and pre-processed variants, we also observe that the latter datasets' increase is slightly faster. This behavior explains why the performance of streaming and neural batch approaches observed in Table 5.4 in the original dataset is similar. The streaming models are unable to take advantage of noticeable changes in data behavior. In contrast, in the pre-processed datasets where these changes are abrupt, the streaming models can adapt their parameters and achieve higher recall values.

Finally, in both evaluation phases, the streaming methods were more efficient in almost all presented recommendation scenarios. ISGD and IBPRMF outperform their batch versions SVD and BPRMF and obtained better results than observed in the neural network approaches. Considering the concept drift and cold start problems, we verify that the incremental ability of the streaming recommender models improved model prediction accuracy.

5.4 Final Considerations

This chapter analyzed batch and stream learning algorithms concerning concept drifts and the cold-start problem. As a by-product of this analysis, a new collaborative filtering supermarket dataset were publicly available alongside two pre-processed variants. As a result of this analysis, it is possible to observe that streaming recommender systems significantly overcome batch approaches. Thus, more effort should be put into tailoring techniques at the intersection between data streams and recommender systems. For instance, streaming recommender systems were especially beneficial with the cold-start issue and overcame complex neural network approaches in weekly recurrent concept drifts with statistical significance. Next, Chapter 6 presents the proposed optimization variants of four well-known optimizers designed for the streaming learning scheme.

6

Contribution II - Incremental Specialized and Specialized-Generalized Matrix Factorization Models based on Adaptive Learning Rate Optimizers

Most proposed recommendation techniques operate in a batch fashion. In such scenarios, given a training set comprising user-item interactions, a static model is generated and later used for recommendations (BABÜROGLU; DURMUSOGLU; DERELI, 2021). However, the world is dynamic, and it is realistic to assume that interactions between users and items may become available over time (RABIU et al., 2020). Consequently, user preferences may change over time (concept drift), new items can be made available (cold start), or even the system rules can be modified (e.g., clearance sales and actions in periods of the year, month, or week) (VINISKI et al., 2021). Thus, developers of such systems must also be aware of concept drift and cold start issues, which require recommender systems to work incrementally, continuously, and consistently to detect and adapt to changes in data so that performance is not jeopardized (LAGHMARI; MARSALA; RAMDANI, 2018).

As changes in the data stream do not occur in all dataset users and items simultaneously, we must consider adapting the recommender model learning schemes in a specialized manner. However, specialization can be ensured in ad-

dition to user or item parameter learning process. In practice, the definition of how specialization should take place includes analyzing the definition of particular step sizes for each user and item in the dataset and adjusting the convergence according to the recommender performance and individual parameters learned.

In this chapter, we propose adaptive optimization strategies for well-known incremental matrix factorization (MF) methods. Our approaches derive from RMSprop (TIELEMAN; HINTON, 2012), Adam (KINGMA; BA, 2015), AMSGrad (REDDI; KALE; KUMAR, 2018), and Nadam (DOZAT, 2016) optimizers, which are adaptive gradient descent-based algorithms often used in a batch fashion. The proposed method differs from the MLF (LUO et al., 2021) optimizer in terms of the use of a single additional array, as it specifies the target of personalization: the user or item. Furthermore, the main purpose is to apply the personalization in an incremental scenario, in which the models were adapted according to changes in data and with positive-only feedback data.More specifically, our contribution is three-fold:

- An analysis of the optimizers application in an incremental fashion as part of matrix factorization algorithms.
- The proposal of optimizer variants that include both user and item specialized and specialized-generalized incremental methods. These optimizers are also substituted into existing MF algorithms.
- An empirical analysis that demonstrates that adaptive learning rate optimizers induce better adjustment in the MF model's parameters and, consequently, more accurate recommendations.

The remainder of this Section is organized as follows. Section 6.1 describes popular adaptive learning rate optimizers. Section 6.2 details the proposed optimization methods and how they are coupled with the existing incremental MF models. Section 6.3 describes the experimental protocol used to perform the proposed analysis of recommender models. Section 7.6 presents the discussion of the obtained results. Finally, Section 6.5 concludes the paper and describes future work.

6.1 Adaptive Learning Rate Optimizers

This section introduces existing adaptive learning rate optimizers available in the literature.

Most machine-learning methods can be formulated as optimization problems to determine the extremum of an objective function. Accordingly, the three vital steps of a machine learning algorithm are (i) building a model hypothesis, (ii) defining the objective function, and (iii) solving the maximum or minimum of the objective function to determine the model parameters. In this sense, the first two steps are modeling problems, and the third step solves the problem using optimization algorithms (SUN et al., 2020).

Therefore, optimization is a mathematical discipline used to solve decision-making problem. The basic idea of optimization is to determine the best solution among the various available options in the given objective function (KASTRATI; BIBA, 2021). Traditionally, optimization research is divided into first-order, high-order, and derivative-free methods (SUN et al., 2020; KASTRATI; BIBA, 2021). First-order optimization methods, known as gradient-based optimization, are primarily based on first-order derivatives or gradient descent algorithms. High-order methods, in turn, are used to address the problem in which an objective function exhibits highly non linear and ill-conditioned behavior. Finally, we can use derivative-free methods in real-world optimization problems, where the derivative of the objective and constraint functions may not exist or is difficult to calculate.

In this study, we focus on traditional MF models, in which the most frequently used optimization strategy to minimize the objective function is gradient descent-based methods, such as RMSProp, Adam, AMSGrad, Nadam, Momentum, Adadelta, and Adagrad optimizers. However, the following sections introduce the RMSProp, Adam, AMSGrad, Nadam, and gradient descent optimization strategies selected to replace the traditional SGD optimizer in the MF model. We selected RMSProp, Adam, AMSGrad, and Nadam optimizers because they exhibited superior and more robust performance in many works when compared with other optimization techniques (DOGO et al., 2018; YU; LIU, 2019; CHAUDHURY; YAMASAKI, 2021)

RMSprop Optimizer

RMSprop (TIELEMAN; HINTON, 2012) is an unpublished adaptive learning rate method that is similar to the SGD algorithm with momentum. It was developed as a stochastic technique for mini-batch learning, which works well in online and non stationary settings. RMSprop and gradient descent differ

in gradient computation. RMSProp uses the exponential decaying average $\overrightarrow{v_t}$ (Equation 6.1) method to discard the squared values of the gradients distant from the current time step. The algorithm converges rapidly after obtaining convex structure. In RMSprop's update rules depicted in Equation 6.2, the gradients ($\overrightarrow{g}$) are divided by the running average of their recent magnitudes ($\overrightarrow{v_t}$) (TIELEMAN; HINTON, 2012). The γ parameter is the momentum (also called the decay rate) and is usually set to 0.9 (TIELEMAN; HINTON, 2012). Considering that the value of $\overrightarrow{v_t}$ may converge to 0, weights can "blow up". To prevent the gradients from blowing up, RMSprop includes a padding factor ϵ in the denominator, often $\epsilon = 10^{-8}$, to avoid inconsistent computation (KINGMA; BA, 2015).

$$\overrightarrow{v_t} \leftarrow \gamma \times \overrightarrow{v_{t-1}} + \overrightarrow{g}^2(1-\gamma) \tag{6.1}$$

$$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \overrightarrow{g}\frac{\eta}{\sqrt{\overrightarrow{v_t}} + \epsilon} \tag{6.2}$$

RMSprop deals with gradient vanishing and explosion issues using a moving average of past squared gradients, normalizing the gradient accordingly. This normalization adapts the learning rate, decreasing it for large gradients to avoid exploding and increasing it for small gradients to avoid vanishing (TIELEMAN; HINTON, 2012; NGUYEN; TSILIGIANNI; DELIGIANNIS, 2018).

Adaptive Moment Estimation - Adam

Adaptive moment estimation (Adam) computes adaptive learning rates for each model parameter (KINGMA; BA, 2015). Similar to the RMSprop optimizers, Adam stores an exponentially decaying average of the past squared gradients v_t (Equation 6.3).

$$\overrightarrow{v_t} \leftarrow \beta_2\overrightarrow{v_{t-1}} + (1-\beta_2)\overrightarrow{g}^2 \tag{6.3}$$

Adam also mantains an exponentially decaying average of past gradients m_t, similar to momentum (Equation 6.4).

$$\overrightarrow{m_t} \leftarrow \beta_1\overrightarrow{m_{t-1}} + (1-\beta_1)\overrightarrow{g} \tag{6.4}$$

The m_t and v_t parameters are estimates of the first (the mean) and the second

moments (the uncentered variance) of the gradients, respectively; hence the method is named accordingly. As m_t and v_t are initialized as vectors of 0's, they are biased towards zero, especially during the initial time steps and especially when the decay rates are low (i.e., β_1 and β_2 are close to 1, propose default values of 0.9 and 0.999, respectively). Thus, to counteract these biases, Adam computes the bias-corrected first and second moment estimates using Equations 6.5 and 6.6, respectively, where t is the current time step (epoch).

$$\overrightarrow{\widehat{m}_t} \leftarrow \frac{\overrightarrow{m_t}}{(1-\beta_1)^t} \tag{6.5}$$

$$\overrightarrow{\widehat{v}_t} \leftarrow \frac{\overrightarrow{v_t}}{(1-\beta_2)^t} \tag{6.6}$$

Finally, Adam's optimization update rule in Equation 6.7 uses the first (Equation 6.5) and second-moment (Equation 6.6) estimations as follows:

$$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \frac{\overrightarrow{\widehat{m}_t}}{\sqrt{\overrightarrow{\widehat{v}_t}} + \epsilon} \tag{6.7}$$

AMSGrad Optimizer

The AMSGrad (REDDI; KALE; KUMAR, 2018) algorithm overcomes the problem in which Adam cannot converge to the optimal solution. The AMSGrad optimizer proposes a new parameter update method that uses the maximum of the past squared gradients $\max(\overrightarrow{v_t})$ (Equation 6.8) rather than the exponential average ($\overrightarrow{v_t}$) directly to update the parameters (Equation 6.9). AMSGrad also calculates the exponentially decaying average of past squared gradients ($\overrightarrow{v_t}$ - Equation 6.3) and the exponentially decaying average of past gradients ($\overrightarrow{m_t}$ - Equation 6.4). Therefore, $\max(\overrightarrow{v_t})$ is an additional parameter.

$$\max(\overrightarrow{v_t}) \leftarrow \max\left(\max(\overrightarrow{v_t}), \overrightarrow{v_t}\right) \tag{6.8}$$

$$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \frac{\overrightarrow{m_t}}{\sqrt{\max(\overrightarrow{v_t})} + \epsilon} \tag{6.9}$$

Nesterov-accelerated Adaptive Moment Estimation - Nadam)

The Nesterov-accelerated adaptive moment estimation (Nadam) (DOZAT, 2016) algorithm combines the Adam and nesterov accelerated gradient (NAG). To incorporate NAG into Adam, the authors modified its momentum term $\overrightarrow{m_t}$, thus defining the adjusted momentum ($\overrightarrow{\widehat{m}_{a(t)}}$) and similarly replacing the previous momentum vector with the current momentum vector. To simplify the adjusted moment term ($\overrightarrow{\widehat{m}_{a(t)}}$), Nadam used the corrected bias ($\overrightarrow{\widehat{m}_t}$ - Equation 6.5) of the current moment term.

$$\overrightarrow{\widehat{m}_{a(t)}} \leftarrow \beta_1 \overrightarrow{\widehat{m}_t} + \frac{(1-\beta_1)\,\overrightarrow{g_t}}{1-\beta_1^t} \tag{6.10}$$

This algorithm aims to increase or decrease the decay factor β_1 over time. We can observe the Nadam update rule, as shown in Equation 6.11.

$$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \frac{\overrightarrow{\widehat{m}_{a(t)}}}{\sqrt{\overrightarrow{\widehat{v}_t}} + \epsilon} \tag{6.11}$$

6.2 Proposed Methods

This section introduces the proposed variants for Adam, AMSGrad, Nadam, and RMSProp with adaptive learning rate optimizers that have been previously introduced in Section 6.1. The main purpose of our proposed optimizer variants is to consider the user or item in the data-stream instance to update the optimizer parameters. Such an analysis enables the learning of personalized optimizer terms for each user or item in the data stream. In contrast to the MLF optimizer described in Section 6.1, our proposed method uses a single auxiliary array (per user or item) to personalize optimizer parameters.

The rationale for proposing optimizer variants is to consider the dynamic characteristics of streaming recommendation scenarios. Such dynamism occurs because the incidence of cold-start and concept-drift problems requires recommendation models to adapt their learning schemes to maintain performance. In this sense, as changes in the data behavior are not observed in the entire dataset and are only for a few users or items, we propose updating the optimizer learning steps individually. The idea here is not to implement an explicit drift detector or treat the new users and new items differently because of the

cold-start incidence but, instead, to make the model adaptive according to the acquired performance.

As both optimizers have as parameters the squared decaying average v_t and/or the decaying average m_t, we show the implementation of the AMSGrad optimizer variants owing to its additional parameter $\texttt{max}(v_t)$. Thus, the following algorithms are prototypes for other adaptive learning rate optimizers. Therefore, to implement RMSProp, Adam, and Nadam variants, we must replace the AMSGrad update rules for their update rules and operations accordingly. Algorithm 1 shows the implementation of the original AMSGrad version, which receives as inputs the latent factors vector $\overrightarrow{\theta}$, current gradient of the error $\overrightarrow{g}$, learning rate η, decay rates β_1 and β_2, padding factor to prevent divisions by zero ϵ , squared decaying average $\overrightarrow{v_{(t)}}$, decaying average $\overrightarrow{m_{(t)}}$, and maximum squared decaying average $\max(\overrightarrow{v_{(t)}})$ of the past gradients g. As output, the AMSGrad algorithm returns the updated latent factors vector $\overrightarrow{\theta}$. The Algorithm 1 is responsible for inplace updating of the AMSGrad parameters based on the gradient of the error $\overrightarrow{g}$ and according to their rules in lines 1, 2, and 3 (described in Section 6.1). Finally, line 4 updates the latent factor vector $\overrightarrow{\theta}$ according to the AMSGrad update rules, and line 5 returns the updated vector.

Algorithm 1: Original AMSGrad Optimizer.

Input: $\overrightarrow{\theta}$: latent factors vector, $\overrightarrow{g}$: gradients of the error, η: learning rate, β_1 and β_2: decay rates, ϵ: padding factor to prevent inconsistent computations, $\overrightarrow{v_{(t)}}$: squared decaying average, $\overrightarrow{m_{(t)}}$: decaying average, $\max(\overrightarrow{v_{(t)}})$: maximum squared decaying average.

Output: $\overrightarrow{\theta}$: updated latent factors vector.

1 $\overrightarrow{v_{(t)}} \leftarrow \beta_2 \times \overrightarrow{v_{(t-1)}} + (1 - \beta_2) \times \overrightarrow{g}^2$

2 $\overrightarrow{m_{(t)}} \leftarrow \beta_1 \times \overrightarrow{m_{(t-1)}} + (1 - \beta_1) \times \overrightarrow{g}$

3 $\max(\overrightarrow{v_{(t)}}) \leftarrow \max(\max(\overrightarrow{v_{(t)}}), \overrightarrow{v_{(t)}})$

4 $\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \left(\frac{\overrightarrow{m_{(t)}}}{\sqrt{\max(\overrightarrow{v_{(t)}}) + \epsilon}} \right)$

5 **return** $\overrightarrow{\theta}$

Section 6.2.1 shows the characteristics of the specialized optimization strategy, in which we learn the personalized terms for each user or item in the data stream. In Section 6.2.2, we present specialized-generalized (SG) variants, which

combine traditional and specialized optimizers to update the latent factor vector.

6.2.1 Specialized Optimizer

In the specialized version, we propose learning personalized optimizer parameters for each user u or item i to update MF models. In this sense, we update latent factor vector of user $\overrightarrow{A_u}$ and item $\overrightarrow{B_i}$ considering a specific optimizer term learned for the current user (user-specialized version) or item (item-specialized version) in the data stream.

In Algorithm 2 we present a specialized version o the AMSGrad optimizer. The specialized AMSGrad optimizer has as inputs the latent factors vector $\overrightarrow{\theta}$, current gradient of the error $\overrightarrow{g}$, and personalization target p (which indicates the identifiers of user u or item i, depending on the specialized variant selected). The specialized version also receives the learning rate η, decay rate β_1, decay rate β_2, padding factor ϵ (to prevent divisions by zero), and AMSGrad parameters, that is, specialized squared decaying average $\overrightarrow{v_{p(t)}}$, specialized decaying average $\overrightarrow{m_{p(t)}}$, and specialized maximum squared decaying average $\max(\overrightarrow{v_{p(t)}})$ of the past gradients g. Algorithm 2 then updates the AMSGrad personalized optimizer parameters $v_{p(t)}$ (Line 1), $m_{p(t)}$(Line 2), and $\texttt{max}(v_{p(t)})$ (Line 3) based on the gradients of the error $\overrightarrow{g}$. Finally, we update the latent factor vector $\overrightarrow{\theta}$ (line 4) using the AMSGrad update rules (Equation 6.9) and return the updated vector (line 5).

If the user-specialized version is selected, both the latent factor vectors of user $\overrightarrow{A_u}$ and item $\overrightarrow{B_i}$ are updated using the personalized optimizer parameters learned for user u in the instance. Suppose we have a function called `update_model` that receives the specialized AMSGrad inputs, then user and item latent factors vectors are updated using Equation 6.12, where $\overrightarrow{v_{u(t)}}$, $\overrightarrow{m_{u(t)}}$, and $\max(\overrightarrow{v_{u(t)}})$ represents the specialized AMSGrad parameters learned for the user u.

$$\begin{aligned}\overrightarrow{A_u} &\leftarrow \texttt{update\_model}(A_u, g_u, u, \eta, \beta_1, \beta_2, \epsilon, \overrightarrow{v_{u(t)}}, \overrightarrow{m_{u(t)}}, \max(\overrightarrow{v_{u(t)}}))\\ \overrightarrow{B_i} &\leftarrow \texttt{update\_model}(B_i, g_i, u, \eta, \beta_1, \beta_2, \epsilon, \overrightarrow{v_{u(t)}}, \overrightarrow{m_{u(t)}}, \max(\overrightarrow{v_{u(t)}}))\end{aligned} \tag{6.12}$$

Otherwise, if an item-specialized version is selected, $\overrightarrow{A_u}$ and $\overrightarrow{B_i}$ are updated using the personalized optimizer parameters learned for the item in the instance

Algorithm 2: Incremental Specialized AMSGrad Optimizer

Input: $\overrightarrow{\theta}$: latent factors vector, $\overrightarrow{g}$: gradients of the error, p: user or item identifier based on specialized variant, η: learning rate, β_1 and β_2: decay rates, ϵ: padding factor to prevent inconsistent computations, $\overrightarrow{v_{p(t)}}$: specialized squared decaying average, $\overrightarrow{m_{p(t)}}$: specialized decaying average, $\max(\overrightarrow{v_{p(t)}})$: specialized maximum squared decaying average.

Output: $\overrightarrow{\theta}$: updated latent factors vector.

1 $\overrightarrow{v_{p(t)}} \leftarrow \beta_2 \times \overrightarrow{v_{p(t-1)}} + (1 - \beta_2) \times \overrightarrow{g}^2$

2 $\overrightarrow{m_{p(t)}} \leftarrow \beta_1 \times \overrightarrow{m_{p(t-1)}} + (1 - \beta_1) \times \overrightarrow{g}$

3 $\max(\overrightarrow{v_{p(t)}}) \leftarrow \max(\max(\overrightarrow{v_{p(t)}}), \overrightarrow{v_{p(t)}})$

4 $\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \left(\frac{\overrightarrow{m_{p(t)}}}{\sqrt{\max(\overrightarrow{v_{p(t)}}) + \epsilon}} \right)$

5 **return** $\overrightarrow{\theta}$

using Equation 6.13, where $\overrightarrow{v_{i(t)}}$, $\overrightarrow{m_{i(t)}}$, and $\max(\overrightarrow{v_{i(t)}})$ represents the specialized AMSGrad parameters learned for the item i.

$$\begin{aligned} \overrightarrow{A_u} &\leftarrow \texttt{update\_model}(A_u, g_u, i, \eta, \beta_1, \beta_2, \epsilon, \overrightarrow{v_{i(t)}}, \overrightarrow{m_{i(t)}}, \max(\overrightarrow{v_{i(t)}})) \\ \overrightarrow{B_i} &\leftarrow \texttt{update\_model}(B_i, g_i, i, \eta, \beta_1, \beta_2, \epsilon, \overrightarrow{v_{i(t)}}, \overrightarrow{m_{i(t)}}, \max(\overrightarrow{v_{i(t)}})) \end{aligned} \tag{6.13}$$

Considering that we are analyzing the RMSProp, Adam, AMSGrad, and Nadam optimizers, we have two specializations for each version. As our focus is on incremental scenario, we named the variants with the prefix "In" and used the suffix to represent the specialized version. In this sense, the traditional variants are the InRMSProp, InAdam, InAMSGrad, and InNadam optimizers. The user-specialized variants assume the "User" suffix with InRMSPropUser, InAdamUser, InAMSGradUser, and InNadamUser. Finally, the item-specialized variants, which have "Item" suffix, are InRMSPropItem, InAdamItem, InAMSGradItem, and InNadamItem. Table 6.1 presents the update rules of Adam, AMSGrad, Nadam, and RMSProp optimizer, where p indicates the specialization target, which assumes the user or item identifiers during the model updates.

6.2.2 Specialized-Generalized Optimizer

The proposed specialized-generalized variant combines the original optimizer version with a specialized version. Accordingly, it is necessary to learn both optimizer parameters instead of learning only the general or specialized versions. We combine the optimizer's general and specialized variants by joining their respective update rules. Therefore, we use the sum of the particular parameters of both variants to generate the specialized-generalized version. Equation 6.14 shows the specialized-generalized optimizer update rule in which $\overrightarrow{\theta}$ is the user $\overrightarrow{A_u}$ or item $\overrightarrow{B_i}$ latent factor vector, η is step size, and P_o and P_s are parameters that represent the original and specialized optimizers, respectively. The combination of these parameters enables the gradient to store both the local and global latent factor changes.

$$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta\,(P_o + P_s) \tag{6.14}$$

Algorithm 3 presents the AMSGrad specialized-generalized optimizer variant. The input arguments ($\overrightarrow{\theta}$, $\overrightarrow{g}$, p, η, β_1, β_2, ϵ) and the output ($\overrightarrow{\theta}$) are the same as those of the original and specialized versions. However, the specialized-generalized variant stores both general and specialized squared decaying averages ($\overrightarrow{v_{(t)}}$ and $\overrightarrow{v_{p(t)}}$), decaying averages ($\overrightarrow{m_{(t)}}$ and $\overrightarrow{m_{p(t)}}$), and maximum squared decaying averages ($\texttt{max}(\overrightarrow{v_{(t)}})$ and $\texttt{max}(\overrightarrow{v_{p(t)}})$). However, we must update both the general (Algorithm 3 - lines 1, 2, and 3) and specialized (Algorithm 3 - lines 4, 5, and 6) parameters according to AMSGrad update rule. Finally, to update the latent factor vector $\overrightarrow{\theta}$, the specialized-generalized variant combines general and specialized AMSGrad update rules (line 7), in which the main difference is between the specialized and specialized-generalized versions.

To distinguish the optimizers' variants, we named the specialized-generalized versions with the "SGI" prefix and used the suffixess "User" or "Item" to represent the specialized part of the optimizer. The user-specialized-generalized variants are SGIRMSPropUser, SGIAdamUser, SGIAMSGradUser, and SGINadamUser while the item-specialized variants are SGIRMSPropItem, SGIAdamItem, SGIAMSGradItem, and SGINadamItem. Algorithms 1, 2, and 3 present the execution flow of the variants of the AMSGrad optimizer. Accordingly, if we want to select other optimizers, we have to use their update rule presented in Table 6.1, which shows the latent factor vector $\overrightarrow{\theta}$ update for the specialized and specialized-generalized variants of the Adam, AMSGrad, Nadam, and RMSProp

Algorithm 3: Incremental Specialized-Generalized AMSGrad Optimizer.

Input: $\overrightarrow{\theta}$: latent factors vector, $\overrightarrow{g}$: gradients of the error, p: user or item identifier based on specialized variant, η: learning rate, β_1 and β_2: decay rates, ϵ: padding factor to prevent inconsistent computations, $\overrightarrow{v_{(t)}}$: squared decaying average, $\overrightarrow{m_{(t)}}$: decaying average, $\max(\overrightarrow{v_{(t)}})$: maximum squared decaying average $\overrightarrow{v_{p(t)}}$: specialized squared decaying average, $\overrightarrow{m_{p(t)}}$: specialized decaying average, $\max(\overrightarrow{v_{p(t)}})$: specialized maximum squared decaying average.

Output: $\overrightarrow{\theta}$: updated latent factors vector

$\overrightarrow{v_{(t)}} \leftarrow \beta_2 \times \overrightarrow{v_{(t-1)}} + (1-\beta_2) \times \overrightarrow{g}^2$

$\overrightarrow{m_{(t)}} \leftarrow \beta_1 \times \overrightarrow{m_{(t-1)}} + (1-\beta_1) \times \overrightarrow{g}$

$\max(\overrightarrow{v_{(t)}}) \leftarrow \max(\max(\overrightarrow{v_{(t)}}), \overrightarrow{v_{(t)}})$

$\overrightarrow{v_{p(t)}} \leftarrow \beta_2 \times \overrightarrow{v_{p(t-1)}} + (1-\beta_2) \times \overrightarrow{g}^2$

$\overrightarrow{m_{p(t)}} \leftarrow \beta_1 \times \overrightarrow{m_{p(t-1)}} + (1-\beta_1) \times \overrightarrow{g}$

$\max(\overrightarrow{v_{p(t)}}) \leftarrow \max(\max(\overrightarrow{v_{p(t)}}), \overrightarrow{v_{p(t)}})$

$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \left(\frac{\overrightarrow{m_{(t)}}}{\sqrt{\max(\overrightarrow{v_{(t)}}) + \epsilon}} + \frac{\overrightarrow{m_{p(t)}}}{\sqrt{\max(\overrightarrow{v_{p(t)}}) + \epsilon}} \right)$

return $\overrightarrow{\theta}$

optimizers. Additional operations may be necessary based on each optimizer step, which are presented in Section 6.1.

6.3 Experimental Setup

We compare our proposed MF optimization strategies with and against the SGD (VINAGRE; JORGE; GAMA, 2014) and InMLF (LUO et al., 2021) algorithms, as previously introduced in Chapter 3 . We also used PMF (SALAKHUTDINOV; MNIH, 2007) and IBPRMF (RENDLE et al., 2012) recommender models as baselines.

Additionally, we tested the non-machine learning methods *Top Popular* (TopPop) (CREMONESI; KOREN; TURRIN, 2010b) and random (KRISTOFFERSEN; SHEPSTONE; TAN, 2018) for comparison. TopPop consists of recommending items with the best degree of success among all users, instead of modeling the

Table 6.1: Optimizers update rules based on specialized and specialized-generalized versions.

Specialized	Specialized-Generalized
Adam	
$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \frac{\overrightarrow{\widehat{m}_{p(t)}}}{\sqrt{\overrightarrow{\widehat{v}_{p(t)}}+\epsilon}}$	$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \left(\frac{\overrightarrow{\widehat{m}_{(t)}}}{\sqrt{\overrightarrow{\widehat{v}_{(t)}}+\epsilon}} + \frac{\overrightarrow{\widehat{m}_{p(t)}}}{\sqrt{\overrightarrow{\widehat{v}_{p(t)}}+\epsilon}} \right)$
AMSGrad	
$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \times \frac{\overrightarrow{m_{p(t)}}}{\sqrt{\max(\overrightarrow{v_{p(t)}})+\epsilon}}$	$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \times \left(\frac{\overrightarrow{m_{(t)}}}{\sqrt{\max(\overrightarrow{v_{(t)}})+\epsilon}} + \frac{\overrightarrow{m_{p(t)}}}{\sqrt{\max(\overrightarrow{v_{p(t)}})+\epsilon}} \right)$
Nadam	
$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \times \frac{\overrightarrow{\widehat{m}_{pa(t)}}}{\sqrt{\overrightarrow{\widehat{v}_{p(t)}}+\epsilon}}$	$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \times \left(\frac{\overrightarrow{\widehat{m}_{a(t)}}}{\sqrt{\overrightarrow{\widehat{v}_{(t)}}+\epsilon}} + \frac{\overrightarrow{\widehat{m}_{pa(t)}}}{\sqrt{\overrightarrow{\widehat{v}_{p(t)}}+\epsilon}} \right)$
RMSProp	
$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \overrightarrow{g} \frac{\eta}{\sqrt{\overrightarrow{v_{p(t)}}+\epsilon}}$	$\overrightarrow{\theta} \leftarrow \overrightarrow{\theta} + \eta \times \overrightarrow{g} \times \left(\frac{1}{\sqrt{\overrightarrow{v_{(t)}}+\epsilon}} + \frac{1}{\sqrt{\overrightarrow{v_{p(t)}}+\epsilon}} \right)$

user item relationship using a machine learning method. In contrast, the random approach randomly selects items from a set of unobserved items in the recommendation phase.

The incremental MF recommendation model uses the SGD optimizer in its original version, which is the most frequently used method for minimizing the L2-regularized squared error (Equation 3.2). Therefore, we compared the proposed adaptive optimizer variants as replacements for the traditional SGD optimizer.

The following hyper-parameter values were tested in the MF incremental recommender model: optimizer ∈ {SGD, InMLF, InAdam, InAdamUser, InAdamItem, SGIAdamUser, SGIAdamItem, InAMSGrad, InAMSGradUser, InAMSGradItem, SGIAMSGradUser, SGIAMSGradItem, InNadam, InNadamUser, InNadamItem, SGINadamUser, SGINadamItem, InRMSprop, InRMSpropUser, InRMSpropItem, SGIRMSpropUser, and SGIRMSpropItem} learning rate ∈ {0.01, 0.001, 0.0001}, regularization rate equal to 0.01, latent factors ∈ {10, 20, 40, 60, 80}, batch training epochs equal to 10. We also used these hyper-parameters, except for the optimizer, in the PMF and IBPRMF recommendation models.

The best hyper parameters were selected based on a grid search, which exhaustively generates candidates from the specified grid of parameter values. The entire dataset was used for each combination. In this sense, parameter tuning is not a part of the processing time of the methods.

In this study, each best experimental setting were replicated 10 times, thus, the results depict the average and standard deviation of the recall values. A single random seed were tested for each replication and used it with all optimizers to enable further paired comparisons and application of the statistical significance test.

As this analysis have many optimization strategies for comparison, each experiment were executed several times to envision finding any statistically significant difference. Therefore, each optimizer's best variant was selected and applied non-parametric tests. Such analysis follows the protocol reported in (DEMSAR, 2006) by combining the Friedman (FRIEDMAN, 1937) and Nemenyi post-hoc (NEMENYI, 1963) statistical tests.

The scores reported in the following section for `RECALL@K` and `NDCG@K` were obtained in the test portion of the experiments. As the best set of parameters was selected, which include the number of latent features, the processing time of the recommender models in this study. The greater the number of latent factors f, the longer is the processing time of the recommender models. Furthermore, the proposed specialized and specialized-generalized optimizer variants are more time and memory consuming than the traditional SGD optimizer because of the storage of the parameters for all users or items of the system.

6.4 Results and Analysis

This section presents the results of experiments using four real-world datasets. The results obtained by incremental MF models are presented in Tables 6.2 and 6.3. These tables depict the RECALL@K and NDCG@K (with $K = 10$) obtained by each optimizer along with the analyzed datasets. The proposed optimizers were compared against the ISGD, MF-InMLF, IBPRMF, PMF, TopPop, and Random baselines and marked the best results per dataset in bold. This study also reports the statistical test results obtained by combining the Friedman and Nemenyi tests, assuming a 95% confidence level throughout the discussion. The p-values for each tested dataset are shown in Figure 6.2. We also observed significant differences in computing the critical distance between the analyzed optimizers and baselines (Figure 6.2).

Regarding the baseline methods (Tables 6.2 and 6.3), the MF with InMLF (LUO et al., 2021) optimizer provide the best RECALL@10 and NDCG@10 values in the Amazon Books dataset. Considering the Movie Lens 1M dataset, the ISGD

and IBPRMF models provided the best baseline results. By contrast, for the Movie Tweetings dataset, the ISGD model obtained the highest RECALL values. Finally, in the supermarket dataset (SMDI-200UE), the PMF model showed the best performance.

Table 6.2: RECALL@10 and NDCG@10 values obtained by the incremental MF recommender model for the Amazon Books and Movie Lens 1M datasets.

Models	Amazon Books		Movie Lens 1M	
	RECALL@10	NDCG@10	RECALL@10	NDCG@10
Baselines				
TopPop	0.0040 ± 0.0000	0.0020 ± 0.0000	0.0390 ± 0.0000	0.0190 ± 0.0000
Random	0.0010 ± 0.0000	0.0000 ± 0.0000	0.0030 ± 0.0002	0.0010 ± 0.0001
	0.0130 ± 0.0001	0.0060 ± 0.0001	0.1200 ± 0.0003	0.0600 ± 0.0001
MF-InMLF	0.0180 ± 0.0002	0.0090 ± 0.0001	0.1070 ± 0.0003	0.0520 ± 0.0002
IBPRMF	0.0150 ± 0.0003	0.0080 ± 0.0002	0.1140 ± 0.0008	0.056 ± 0.0004
PMF	0.0050 ± 0.0001	0.0020 ± 0.0001	0.1210 ± 0.0050	0.0600 ± 0.0028
MF with Adam Optimizers variants				
InAdam	0.0080 ± 0.0001	0.0040 ± 0.0001	0.1300 ± 0.0002	0.065 ± 0.0001
InAdamUser	0.0210 ± 0.0025	0.0100 ± 0.0012	0.1310 ± 0.0002	0.066 ± 0.0001
InAdamItem	0.0210 ± 0.0005	0.0100 ± 0.0002	0.0870 ± 0.0012	0.045 ± 0.0006
SGIAdamUser	0.0160 ± 0.0015	0.0080 ± 0.0007	0.1250 ± 0.0003	0.063 ± 0.0002
SGIAdamItem	0.0190 ± 0.0004	0.0090 ± 0.0002	0.1060 ± 0.0017	0.053 ± 0.0009
MF with AMSGrad Optimizers variants				
InAMSGrad	0.0050 ± 0.0002	0.0020 ± 0.0001	0.1310 ± 0.0003	0.0660 ± 0.0002
InAMSGradUser	0.0360 ± 0.0040	0.0170 ± 0.0020	0.1320 ± 0.0003	0.0660 ± 0.0002
InAMSGradItem	0.0110 ± 0.0005	0.0050 ± 0.0002	0.0850 ± 0.0013	0.0430 ± 0.0007
SGIAMSGradUser	0.0270 ± 0.0045	0.0130 ± 0.0023	0.1320 ± 0.0003	0.0660 ± 0.0001
SGIAMSGradItem	0.0110 ± 0.0005	0.0050 ± 0.0002	0.1020 ± 0.0012	0.0510 ± 0.0006
MF with Nadam Optimizers variants				
InNadam	0.0080 ± 0.0001	0.0040 ± 0.0001	0.1310 ± 0.0002	0.0660 ± 0.0001
InNadamUser	0.0190 ± 0.0020	0.0090 ± 0.0009	0.1310 ± 0.0003	0.0660 ± 0.0002
InNadamItem	0.0210 ± 0.0006	0.0100 ± 0.0003	0.0910 ± 0.0013	0.0460 ± 0.0006
SGINadamUser	0.0140 ± 0.0009	0.0060 ± 0.0005	0.1300 ± 0.0004	0.0650 ± 0.0003
SGINadamItem	0.0180 ± 0.0004	0.0090 ± 0.0002	0.1140 ± 0.0024	0.0570 ± 0.0012
MF with RMSProp Optimizers variants				
InRMSProp	0.0170 ± 0.0004	0.0080 ± 0.0002	0.1260 ± 0.0004	0.0630 ± 0.0002
InRMSPropUser	0.0170 ± 0.0008	0.0080 ± 0.0004	0.1280 ± 0.0003	0.0640 ± 0.0002
InRMSPropItem	0.0160 ± 0.0016	0.0080 ± 0.0007	0.1260 ± 0.0002	0.0630 ± 0.0001
SGIRMSPropUser	0.0090 ± 0.0006	0.0040 ± 0.0003	0.1300 ± 0.0003	0.0650 ± 0.0001
SGIRMSPropItem	0.0080 ± 0.0003	0.0040 ± 0.0001	0.1290 ± 0.0005	0.0640 ± 0.0002

Tables 6.2 and 6.3 summarize the performances of the different optimizers analyzed in terms of the RECALL@10 and NDCG@10 values. The InAMSGradUser optimizer obtained the best results across all tested datasets. Considering the RECALL@10 values of InAMSGradUser, the adaptive optimizer increases by up to 2.3 percentage points in the Amazon Books, 1.2 percentage points for the Movie Lens 1M (Table 6.2), 8.2 percentage points for the Movie Tweetings, and 11.1 percentage points for the SMDI-200UE dataset (Table 6.3), compared with the traditional SGD baseline. Furthermore, compared with adaptive InMLF optimization, the InAMSGradUser optimizer provided an increase by up to 1.8, 2.5, 13.1, and 14.7 percentage points for the Amazon Books, Movie Lens

Table 6.3: RECALL@10 and NDCG@10 values obtained by the incremental MF recommender model for the Movie Tweetings and SMDI-200UE datasets.

Models	Movie Tweetings		SMDI-200UE	
	RECALL@10	NDCG@10	RECALL@10	NDCG@10
Baselines				
TopPop	0.0070 ± 0.0000	0.0030 ± 0.0000	0.1500 ± 0.0000	0.0870 ± 0.0000
Random	0.0000 ± 0.0000	0.0000 ± 0.0000	0.0020 ± 0.0001	0.0010 ± 0.0000
SGD	0.1510 ± 0.0032	0.0630 ± 0.0014	0.2290 ± 0.0004	0.1330 ± 0.0002
InMLF	0.1010 ± 0.0020	0.0430 ± 0.0008	0.1930 ± 0.0023	0.1080 ± 0.0012
IBPRMF	0.0580 ± 0.0004	0.0280 ± 0.0003	0.2380 ± 0.0013	0.1380 ± 0.0008
PMF	0.0090 ± 0.0015	0.0050 ± 0.0009	0.3390 ± 0.0058	0.2250 ± 0.0030
MF with Adam Optimizers variants				
InAdam	0.1660 ± 0.0013	0.074 ± 0.0006	0.3210 ± 0.0004	0.2030 ± 0.0003
InAdamUser	0.2250 ± 0.0045	0.1110 ± 0.0027	0.3370 ± 0.0004	0.2010 ± 0.0017
InAdamItem	0.2070 ± 0.0016	0.0990 ± 0.0012	0.1950 ± 0.0026	0.1210 ± 0.0021
SGIAdamUser	0.2060 ± 0.0064	0.1010 ± 0.0034	0.3290 ± 0.0011	0.1840 ± 0.0014
SGIAdamItem	0.2120 ± 0.0010	0.1000 ± 0.0007	0.2100 ± 0.0042	0.1290 ± 0.0023
MF with AMSGrad Optimizers variants				
InAMSGrad	0.1510 ± 0.0029	0.0720 ± 0.0014	0.3140 ± 0.0011	0.1820 ± 0.0011
InAMSGradUser	0.2320 ± 0.0028	0.1150 ± 0.0017	0.3400 ± 0.0005	0.2080 ± 0.0007
InAMSGradItem	0.2090 ± 0.0023	0.1 ± 0.0013	0.2350 ± 0.0039	0.1560 ± 0.0028
SGIAMSGradUser	0.2090 ± 0.0025	0.102 ± 0.0013	0.3380 ± 0.0003	0.2010 ± 0.0003
SGIAMSGradItem	0.2090 ± 0.0011	0.101 ± 0.0007	0.2550 ± 0.0030	0.1660 ± 0.0025
MF with Nadam Optimizers variants				
InNadam	0.1570 ± 0.0004	0.0700 ± 0.0002	0.3260 ± 0.0006	0.1970 ± 0.0004
InNadamUser	0.2280 ± 0.0033	0.1120 ± 0.0018	0.3400 ± 0.0006	0.2010 ± 0.0018
InNadamItem	0.2190 ± 0.0012	0.1050 ± 0.0009	0.2050 ± 0.0051	0.1280 ± 0.0029
SGINadamUser	0.2040 ± 0.0048	0.0990 ± 0.0026	0.3280 ± 0.0015	0.1820 ± 0.0014
SGINadamItem	0.2220 ± 0.0012	0.1060 ± 0.0009	0.2240 ± 0.0073	0.1360 ± 0.0043
MF with RMSProp Optimizers variants				
InRMSProp	0.1780 ± 0.0024	0.0910 ± 0.0015	0.1960 ± 0.0004	0.1170 ± 0.0004
InRMSPropUser	0.1770 ± 0.0021	0.0930 ± 0.0015	0.1940 ± 0.0010	0.1190 ± 0.0005
InRMSPropItem	0.1710 ± 0.0043	0.0880 ± 0.0023	0.2030 ± 0.0032	0.1220 ± 0.0016
SGIRMSPropUser	0.1880 ± 0.0026	0.0960 ± 0.0014	0.3130 ± 0.0013	0.1610 ± 0.0016
SGIRMSPropItem	0.1650 ± 0.0012	0.0850 ± 0.0007	0.3120 ± 0.0014	0.1640 ± 0.0015

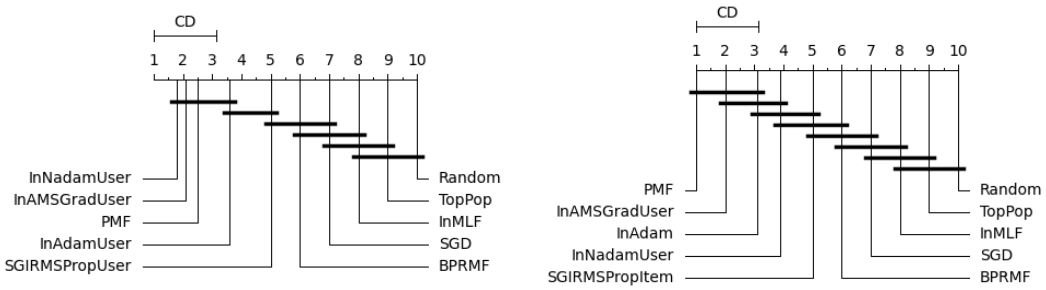
1M, Movie Tweetings, and SMDI-200UE datasets, respectively. Concerning the NDCG@10 metric, the results showed a lower increase, which is natural because the NDCG evaluates the item ranking in the TOP@10 ranked list. Regarding the InAMSGradUser and SGD NDCG@10 values, we observed increases of up to 1.1, 0.6, 5.2, and 7.5 percentage points for the Amazon Books, Movie Lens 1M, Movie Tweetings, and SMDI-200UE datasets, respectively. However, although the increase in NDCG values was smaller than that in RECALL, there were also significant differences between the results.

The results showed that the PMF model provided competitive results for the Movie Lens and SMDI-200UE datasets. Moreover, considering the Amazon Books and Movie Tweeting datasets, which are the largest amog the tested datasets, the results are not significantly different from the TopPop baseline. As observed in the incremental MF model results (Tables 6.2 and 6.3), considering the optimizer variants, the user-specialized version provided superior

performance in most cases. In all the analyzed datasets, the InAdamUser, InAMSGradUser, and InNadamUser variants showed the best results compared to the other variants of each optimizer. For the RMSProp optimizer, the user specialized-generalized version presented the best results. In contrast, in most experiments item-specialized and item-specialized-generalized variants did not significantly differ from the SGD baseline. I can explain this behavior by searching for datasets of distinct users and items, where the number of users is higher than the number of items. Additionally, as the focus is the collaborative filtering scenario, where the primary purpose is to model and learn the relationship between users, it makes sense to learn individualized optimizer parameters for each user, because the optimizer also uses these parameters to update the latent factor vector of the items.

Considering the statistical significance test, although the InAMSGradUser rates were greater, some optimizers did not show significant differences considering the Nemenyi test with a 95% confidence level. In the Movie Lens 1M dataset, the InAMSGradUser optimizer did not show a significant difference from the SGIAMSGradUser, InNadamUser, InAdamUser, InAMSGrad, InNadam, and InAdam optimizers. Considering the Movie Tweetings dataset, the optimizers without significant differences from InAMSGradUser are InNadamUser, InAdamUser, SGINadamItem, InNadamItem, and SGINadamItem. Finally, in the SMDI-200UE dataset, the InAMSGradUser optimizer does not differ from InNadamUser, SGIAMSGradUser, InAdamUser, SGIAdamUser, SGINadamUser, and InNadam.

Figure 6.1 shows the critical distance (CD) of the best variants of each optimizer, considering the obtained results for all datasets. The general results concerning the RECALL@10 values demonstrate the best performance of user-specialized variants. In addition, by comparing the RECALL@10 values across all the analyzed datasets, it is possible to confirm that the AMSGradUser and InNadamUser optimizers presented the best performance in the experiments, showing significant differences from the other optimizers, except for the PMF model in the SMDI-200UE dataset, which presented competitive results. Furthermore, considering the NDCG@10 values, the PMF model outperformed the other optimizers. On the other hand, in contrast to what was observed in the RECALL@10 analysis, regarding the NDCG@10 values, the InAdam and SGIRMSPropItem optimizers outperform the InAdamUser and SGIRMSPropIUser variants.



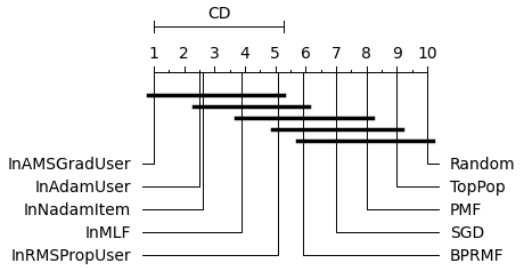
(a) RECALL@10.

(b) NDCG@10.

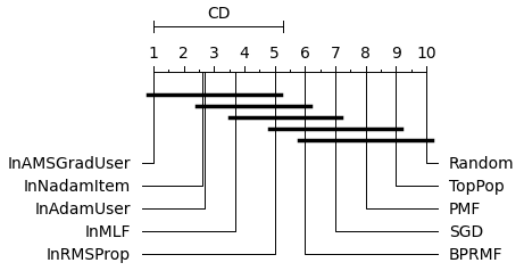
Figure 6.1: Nemenyi test results on RECALL@10 and NDCG@10 values in the analysed datasets.

Figure 6.2 presents the results of Friedman and Nemenyi's statistical significance test, considering the experiments of all techniques in each analyzed dataset. The legends present the Friedman p-values and show significant differences, assuming a 95% (p-value < 0.05) confidence level for the results for all datasets. Therefore, the graph of the critical distance obtained by the Nemenyi test shows the rankings and significant differences of each method. Figure 6.2 presents the statistical significance tests, in that the behavior of the results is similar to that presented in Tables 6.2 and 6.3. The user-specialized variants provided the best results in most cases and outperformed the specialized-generalized versions of the AMSGrad, Nadam, and Adam optimizers. InAMSGradUser and InAdamUser proposed specialized optimizer variants provide the best results for all analyzed datasets considering RECALL@10. Regarding the NDCG@10 values, only for the SMDI-200UE dataset, InAdamUser does not appear as the best variant of the Adam Optimizer. However, considering the RMSProp optimizer, the best results were obtained in the user-specialized-generalized version (SGIRMSPropUser). The results presents different behaviors in the Amazon Books dataset results, in which the item-specialized Nadam optimizer variant (InNadamItem) presented the best performance compared to the other versions.

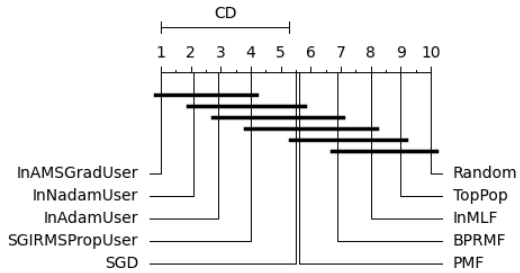
In contrast to the results obtained in the combined statistical analysis of all datasets (Figure 6.1), considering the results in every single dataset (6.2), the PMF model presented the best results only for the SMDI-200UE dataset. However, in the general analysis, the PMF obtained the best performance because the RECALL@10 and NDCG@10 values in the SMDI-200UE dataset were higher than those of the other datasets.



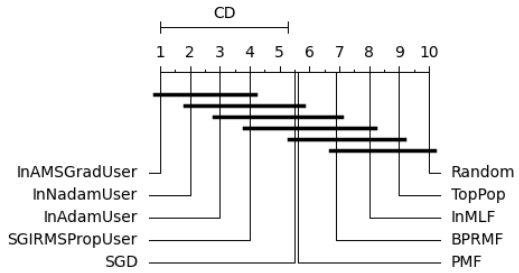
(a) Amazon Books RECALL@10 (p-value = 2.64×10^{-15}).



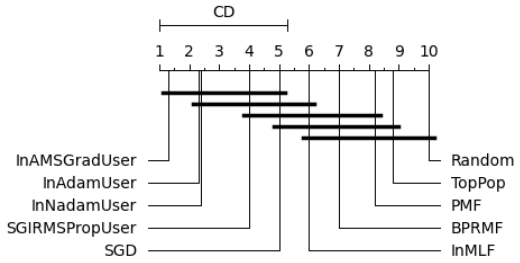
(b) Amazon Books NDCG@10 (p-value = 3.07×10^{-15}).



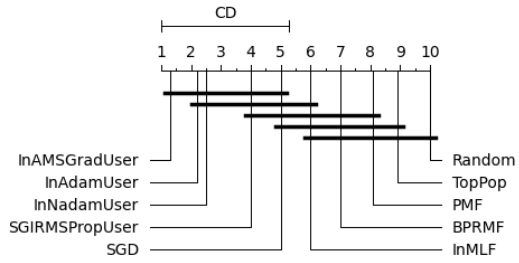
(c) Movie Lens RECALL@10 (p-value = 2.64×10^{-15}).



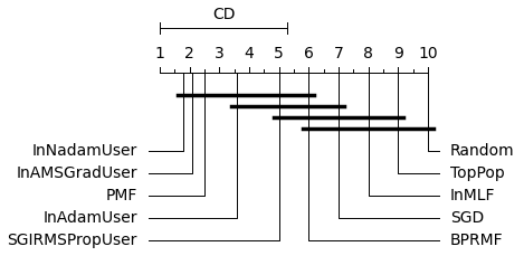
(d) Movie Lens NDCG@10 (p-value = 2.41×10^{-15}).



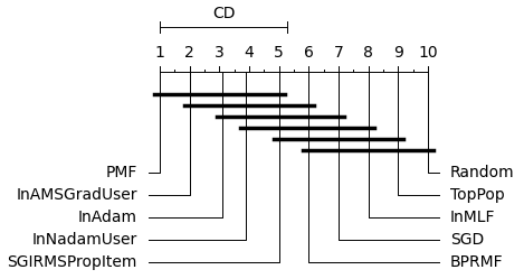
(e) Movie Tweetings RECALL@10 (p-value = 3.60×10^{-15}).



(f) Movie Tweetings NDCG@10 (p-value = 3.29×10^{-15}).



(g) SMDI-200UE RECALL@10 (p-value = 7.9×10^{-15}).



(h) SMDI-200UE NDCG@10 (p-value = 1.78×10^{-15}).

Figure 6.2: Critical distances of the Nemenyi test for tested datasets. All p-values refer to the Friedman test.

Comparing the obtained results across the analyzed datasets, the proposed user-specialized and user-specialized-generalized variants provided superior

performance to the tested baselines in most cases. Additionally, combining the adaptive learning rate optimizers in the MF model, Adam, Nadam, AMSGrad, and RMSProp significantly increased the RECALL and NDGC rates compared with the traditional SGD baseline.

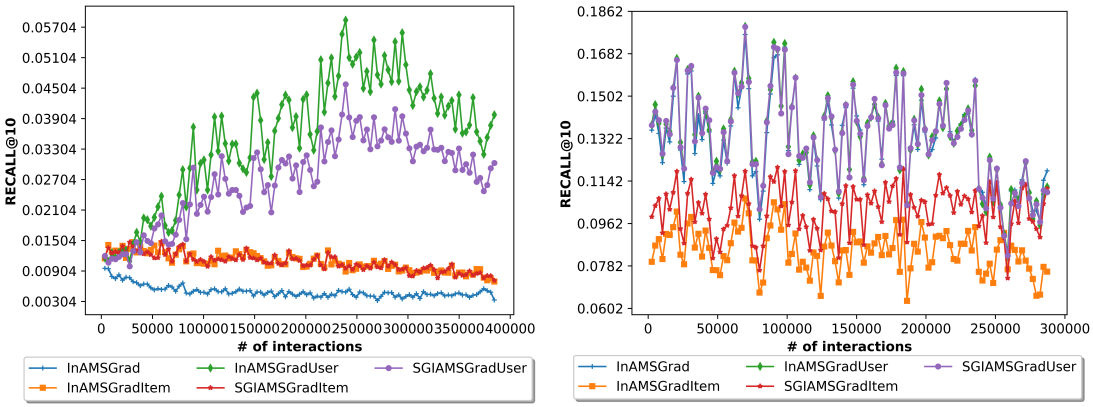
6.4.1 Streaming Analysis of the Results

As the focus of this thesis is the streaming scenario, it is essential to evaluate the tested methods by considering the evolution of the results according to the emergence of new instances $\langle u, i \rangle$in the stream. Figure 6.3 presents the windowed evaluation of RECALL@10 values by analyzing the best optimizer and their variants in each dataset. Were considered a window with a size 5% of the number of test interactions. Considering the Amazon Books dataset results (Figure 6.3a), the AMSGradUser and SGIAMSGradUser optimizers outperform other AMSGrad variants during the entire streaming test set. However, the specialized version AMSGradUser maintains the best windowed RECALL@10 values during the data stream fashion. The item-specialized (InAMSGradItem) and item-specialized-generalized (SGIAMSGradItem) variants showed similar results and presented superior results to the traditional InAMSGrad optimization strategy.

Concerning the results obtained in the Movie Lens dataset experiments (Figure 6.3b), we can confirm the reduced differences in the RECALL@10 values compared to the windowed evaluation results with the obtained average RECALL@10 rates (Table 6.2). The both traditional (InAMSgrad), user-specialized (InAMSgradUser), and user-specialized-generalized (SGIAMSgradUser) variants of the AMSGrad optimizer show similar behavior in the plotted results. Different from what were obtained in the Amazon Books dataset, in the Movie Lens dataset, the item-specialized (InAMSGradItem) and item-specialized-generalized (SGIAMSGradItem) variants of AMSGrad optimizer presented inferior results than the traditional InAMSGrad version.

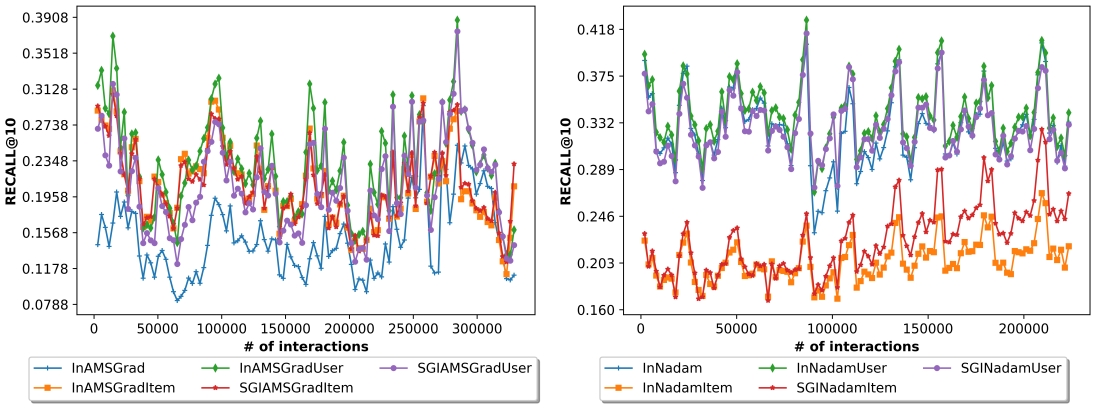
Figure 6.3c shows the windowed RECALL@10 values obtained using the AMSGrad optimizer variants in the movie tweeting dataset. This analysis shows that the proposed AMSGrad variants outperform the traditional optimizer. The user-specialized optimizer (InAMSGradUser) provided superior performance in some regions of the graph, as proven by the average RECALL@10 rates (Table 6.3), in which we obtained a RECALL@10 value of 23.20% using the InAMS-

GradUser optimizer. For InAMSGradItem, SGIAMSGradUser, and SGIAMSGradUser optimizers, the obtained RECALL@10 value was 20.90%.



(a) AMSGrad optimizers in the Amazon Books dataset.

(b) AMSGrad optimizers in the Movie Lens dataset.



(c) AMSGrad optimizers in the Movie Tweetings dataset.

(d) Nadam optimizers in the SMDI-200UE dataset.

Figure 6.3: Windowed evaluation of RECALL@10 values obtained by the best optimizer and their variants in each dataset (We consider a window size of 5% of the number of interactions for each dataset test portion).

Finally, considering the SMDI-200UE dataset results, Figure 6.3d shows the results for the Nadam optimizer variants. A similar behavior were observed from the obtained results in the Movie Lens dataset, in which the traditional (InNadam), user-specialized (InNadamUser), and user-specialized-generalized (SGINadamUser) variants presented similar results. Furthermore, the item specialized (InNadamItem) and item-specialized-generalized (SGINadamItem) versions also provided worse results than the traditional version. Nonetheless, although the windowed evaluation presented similar results, InNadamUser and

SGINadamUser outperformed the InNadam optimizer in some regions of the test data stream.

The obtained results in the analyzed datasets could be justified by comparing the size (number of instances) and number of distinct users and items. The Movie Lens and SMDI-200UE datasets had fewer users and items than the amazon books and movie tweeting datasets. In this sense, the proposed optimizer variants provided the best performance in datasets with a higher number of users and items, which is reasonable because individual optimizer parameters were trained for each available user or item in the stream. The amazon books and movie tweeting datasets also have more instances than the others, showing that our methods work well in large datasets with a high incidence of new users and items.

6.5 Conclusion

This study proposed novel adaptive learning rate optimizers for incremental MF recommender systems. Our variants consider user (user-specialized and user-specialized-generalized) or item (item-specialized and item-specialized-generalized) identifiers to model and learn optimizer parameters for each user or item, respectively. Four adaptive optimizers were selected, Adam, Nadam, AMSGrad, and RMSProp, and coupled them with the proposed optimizers. The analysis tested the proposed approaches on three real-world datasets and compared their results with those of a traditional SGD optimizer. As the result of this analysis showed that the InAMSGradUser and InNadamUser variants, which are user-specialized versions of the AMSGrad and Nadam optimizers, significantly outperform the SGD among all datasets, increasing the RECALL@10 values by up to 11.1 percentage points and NDCG@10 by up to 7.5 percentage points. In this sense, combining adaptive learning rate optimizers in the MF model provides more accurate recommendations to users because adapting the learning rates during models' incremental updates makes the models adaptive to changes in data behavior. Next, Chapter 7 introduces the proposed negative-relevant sampling strategies combined with pairwise and point-wise models for the OCCF scenarios.

7

Contribution III - Improving Negative Items Sampling in Streaming Scenarios

Recently, companies in all sectors have widely used recommendation systems and have proven effective in recommending personalized items to users, boosting businesses, and facilitating decision-making processes (LI et al., 2017a; ZHANG et al., 2019). Collaborative Filtering (CF) is one of the most widely used approaches for recommender systems learning and modeling (VOLKOVS; YU, 2015; ZHANG et al., 2013). It leverages the user-item preference, rating, or behavior patterns provided by the systems from a large amount of historical data to make the recommendation. Different recommendation scenarios could result in different CF models (ZHANG et al., 2013).

In practice, most information systems only catch the user's implicit feedback, the available information to learn a recommender system. Implicit feedback scenarios often represent the user interaction with the system and include clicks, playing songs, watching movies, reading books/articles, purchases, and so on (RENDLE; FREUDENTHALER, 2014). Recent research on recommendations has shifted from explicit ratings to implicit feedback (DING et al., 2018), which is much easier to collect compared to the explicit feedback (such as ratings) on item utility (PAN; LIU; MING, 2016; DING et al., 2020).

Despite the easiness of data acquisition, implicit feedback scenarios have specific problems, such as negative feedback's unavailability. The absence of

user negative feedback is called One-Class Collaborative Filtering (OCCF) or positive-only feedback (PAN et al., 2008). Despite the lack of negative feedback, algorithms tailored for OCCF require strategies to assume the unknown relations between users and items as negative (YU; BILENKO; LIN, 2017). There are two ways to incorporate unknown inputs into the model's training: (i) consider that all missing interactions between users and items are negatives, or (ii) select a sub-sample of these missing interactions as negative.

Furthermore, a user listening to a song, watching a movie, browsing a web page, or clicking on a product offer does not necessarily mean that the user likes the corresponding item. Thus, measuring the degree of preference for such interactions is impossible. Consequently, more than the user's interaction records might be required to learn and model their valid preferences (PAN; LIU; MING, 2016). In contrast to multi-class Collaborative Filtering (MCCF) scenarios, which often focus on the observed relations only, special treatments are needed for the missing/unobserved examples in OCCF (YAO et al., 2019). Considering the significant number of missing interactions, it is unfeasible to assume all as negative (RENDLE; FREUDENTHALER, 2014) due to the computational cost.

Due to the absence of negative feedback in the positive-only datasets, some models select unknown observations to serve as negative ones and improve the model's convergence. The BPR (RENDLE et al., 2012) is the most common method to treat positive-only recommendation scenarios, which assumes that an observed interaction should be ranked with a higher score than its unobserved counterparts. Neural Network methods (HE et al., 2017) also provided proper strategies to learn recommender models from binary implicit feedback.

In this Chapter, we introduced the SBRG and MBRG strategies. This approach combines negative and relevant sampling in pairwise (IBPRMF) and point-wise (PMF) models seeking to improve the performance of the methods.

7.1 Background - Ranking-Based Recommender Models

To deal with the implicit recommendation scenarios, we use the Top-N recommendation tasks, which produce a ranking of the N most relevant items to the user for each recommendation interaction. The Top-N CF recommendation techniques aim to generate an optimal item rank function based on a given

context, i.e., each user profile (ZHANG et al., 2013).

Ranking-based recommender models are powerful tools for solving the Top-N recommendation task, which can be optimized pairwise or point-wise. The pairwise method usually has three inputs, i.e., a user u and a positive-negative item pair (i, j). The goal is to maximize the user reference margin between the positive and negative items. Suppose the user preference is estimated from a function $f(\cdot)$ (predictive model), then the objective function (optimization target) is represented by Equation 7.1 (YANG et al., 2021).

$$L_1(O) = -\sum_{(u,i,j)\in O} \log \sigma \left(f(u,i) - f(u,j) \right), \tag{7.1}$$

where σ is the sigmoid function to convert the margin to a probability and to avoid trivial solutions. $O = \{(u,i,j)|i \in I_u^+, j \in I \setminus I_u^+\}$ represents the dataset instances. I is the whole item set, and I_u^+ indicates the set of items the user u interacted with. (DING et al., 2020; YANG et al., 2021). For the point-wise method, each instance is a user-item pair (u,i), and the item i receives both positive and unobserved items for the user u. The user preference estimation is treated as a classification problem, where f is optimized by the cross entropy objective (Equation 7.2).

$$L_2(O) = -\sum_{(u,i)\in O^+} \log \left(f(u,i) \right) - \sum_{(u,i)\in O^-} \left(1 - \log \left(f(u,i) \right) \right), \tag{7.2}$$

where $O^+ = \{(u,i)|i \in I_u^+\}$ and $O^- = \{(u,i)|i \in I \setminus I_u^+\}$ are the set of positive and negative samples. $O = O^+ \cup O^-$ denotes the complete dataset instances.

7.2 Negative Sampling Approaches in Streaming Scenarios

As presented in Session 7.1, pairwise and point-wise models in the Top-N tasks must select unobserved items during the model's training/update to serve as negative items. The common approach is to apply a uniform negative sampling (Session 7.2.1). However, many works suggest that uniform negative sampling needs to provide better examples for model training/update.

7.2.1 Uniform Random Sampling

The uniform random sampling is an option that selects the negative item j from the entire set of unobserved items to the user u ($\{j \in I \setminus I_u^+\}$). Additionally, both selected items often received the same selection probabilities during model updates. Algorithm 4 shows the uniform negative sampling used in a pairwise model update.

Algorithm 4: Uniform Negative Sampling for a pairwise model

Data: Set of positive observations O^+, set of all items I
Input: k: number of negative items per positive one

```
foreach (u, i) ∈ O+ do
    x ← 0
    while x < k do
        draw j from I \ I_u+   ▷ sampling j from the set of unobserved items
        model_update(u, i, j)   ▷ learn parameters using the three inputs (u, i, j)
        x ← x + 1
    end
end
```

7.3 Proposed Strategies to Candidate Items Sampling

This section presents our proposed candidate item sampling strategies to improve the accuracy of the streaming recommendation models. Our methods combine the negative selection from the unobserved items set with the selection of items that could be interesting for users. Section 7.3.1 introduces the Similarity-based Negative-Relevant Sampling (SBNRS), and Section 7.4 presents the Model-based Negative-Relevant Sampling. Both strategies are used for generating the possible relevant items set for each user (I_u^r), i.e., such items that could be of interest to a user based on knowledge acquired during the evolution of the model. Concerning each user's generated set of possible relevant items (I_u^r), we aim to adjust the pairwise and point-wise models to consider both unobserved and relevant items during training/update. In this sense, instead of only selecting items to be trained as negatives from the entire set of unobserved items, the idea is to choose candidates from the relevant items set I_u^r.

7.3.1 Similarity-based Relevant Items Set Generation

This section presents the proposed Similarity-based Relevant Generation (SBRG) and introduces the steps to generate the relevant items set. The cosine function, defined as the inner product of two vectors divided by the product of their lengths (YE, 2011), is the most popular and widely used similarity measure among the existing similarity measures. Its calculation is efficient, especially for sparse vectors, as only the non-zero dimensions are considered (LI; HAN, 2013). This characteristic is significant in the implicit feedback scenarios given the sparsity in the interaction matrix R. Given two m-dimensional vectors $\vec{v}$ and $\vec{w}$, where m is the number of users, the Cosine similarity between them is calculated as follows:

$$\text{Cosine}(\vec{v}, \vec{w}) = \frac{\vec{v} \bullet \vec{w}}{\|\vec{v}\| \, \|\vec{w}\|} = \frac{\sum_{i=0}^{n} \vec{v}_i \times \vec{w}_i}{\sqrt{\sum_{i=0}^{n} \vec{v}_i{}^2} \sqrt{\sum_{i=0}^{n} \vec{w}_i{}^2}} \tag{7.3}$$

As we work in a streaming scenario with implicit feedback, computing the full item-item similarity matrix for every interaction emerging from the stream is infeasible in practice. Thereby, considering the current instance (u, i), we only update the similarity values between the item i and the other items in I_u^+, i.e., the items that user u has interacted with. The Cosine approach applies the Cosine similarity function to all item-item $(\vec{v}, \vec{w})$ pairs. As a result, Algorithm 5 presents the `update_similarity` function steps to update similarity matrix $S^{n \times n}$, where n is the number of items.

Algorithm 5: Updating the similarity structure incrementally - `update_similarity`

Data: I_u^+: positive items for a user u, S: similarity matrix between items, K: the set of users who interacted with each item, N_r: number of the most similar items to be stored for each user

Input: (u,i), which represents the user u and the item i in the current instance.

1 $I_u^+ \leftarrow I_u^+ \cup i$
2 $K_i \leftarrow K_i \cup u$
3 **foreach** $j \in I_u^+$ **do**
4 $\quad S_{i,j} \leftarrow \frac{K_i \cap K_j}{\|K_i\| \cdot \|K_j\|}$ ▷ Updating the similarity between i and all others items in I_u^+
5 **end**

With the similarities between the available items, the next step is the analysis of possible relevant items for each user. Thus, we retrieve the items most similar to those the user interacted with. Algorithm 6 presents the steps to retrieve the most relevant items to a given user u. Algorithm 6 uses the set of user u positive items I_u^+, the items-items similarity matrix S, and the number of relevant items that we stored for each user N_r. Algorithm 6 receives the user-item pair that represents the dataset instance (u, i) as input. First, we must update the user u set of positive items with incoming item i (line 1) and update the similarity structure based on the item in the stream (line 2). Next, we get a partial similarity matrix $P^{x \times n}$, where x is the number of user u positive items ($x \leftarrow |I_u^+|$) and n is the number of items (Algorithm 6 - Line 3). To retrieve the most similar items to those the user interacted with, we sum the similarities along the positive items, generating a vector V containing the sum of similarities for each available item in I (Algorithm 6 - Line 4). Naturally, considering that V also have the user positive items (I_u^r) and their similarities values are higher, the next step is to remove them from V (Algorithm 6 - Line 5) and then sort V in ascending order (Algorithm 6 - Line 6). Finally, we can recover the N_r most similar items by selecting N_r items from the end of the vector V (line 7).

Algorithm 6: SBRG strategy - `update_memory`

Data: I_u^+: positive items for a user u, S: similarity matrix between items, N_r: number of the most similar items to be stored for each user

Input: (u,i), which represents the user u and the item i in the current instance.

Output: I_u^r

1 $I_u^+ \leftarrow I_u^+ \cup i$

2 `update_similarity(u,i)` ▷ Algorithm 5

3 $P \leftarrow D^{a \times b}$, such that $a \in I_u^+$ and $b \in I$, and $d_{a,b} = S_{a,b}$ ▷ get a partial similarity matrix with the weight vectors of items in I_u^+

4 $V \leftarrow \sum_{k=0}^{a} P_{k,b}$ ▷ sum of similarities considering items in I_u^+

5 $V \leftarrow V \backslash I_u^+$

6 Sort V in ascending order

7 $I_u^r \leftarrow V_k$, such that $(|V| - N_r \leq k \leq |V|)$

7.4 Model-based Relevant Items Set Generation

This section introduces the proposed Model-based Relevant Generation (MBRG) strategy. Our model-based sampler is more straightforward than the SBRG strategy since we do not need to store the similarity structure during the recommendation process. Algorithm 7 presents the MBNRS steps for generating the set of possible relevant items to a user. MBNRS uses the user u latent factors vector A_u, the items factor Matrix B, and the number of relevant items to store for each user N_r. As input, the MBNRS strategy receives the user identifier u and item i identifiers and returns the updated set of relevant items I_u^r as output. The main focus of the MBNRS strategy is to consider the learning model in the sampling schema. Therefore, as we are working with MF models, we use the dot product between the user's latent factors vector A_u and the transpose item's latent factor matrix B^T to acquire all available items predictions V for the respective user u (Algorithm 7 - Line 2). Next, we remove the user u positive items predictions from V (Algorithm 7 - Line 3) and sort V in ascending order (Algorithm 7 - Line 4). Considering those the higher item predictions are at the end of V, we select the last N_r items from V to serve as the most relevant items I_u^r to user u (Algorithm 7 - Line 5). Finally, Function `update_memory` returns the updated set of relevant items I_u^r (Algorithm 7 - Line 6).

Algorithm 7: MBRG strategy - `update_memory`

Data: A: user's latent factor vectors, B: item's latent factor vectors, N_r: number of the most similar items to be stored for each user

Input: (u, i), where u represents the user identifier and i the item identifier.

Output: I_u^r: set of relevant items to user u.

1 $I_u^+ \leftarrow I_u^+ \cup i$ ▷ Updating user u set of positive items.
2 $V \leftarrow A_u \cdot B^T$ ▷ Get the predictions of all items to user u
3 $V \leftarrow V \backslash I_u^+$ ▷ removing the user positive items I_u^+ from V
4 Sort V in ascending order
5 $I_u^r \leftarrow V_k$, such that $(|V| - N_r \leq k \leq |V|)$
6 **return** I_u^r

7.4.1 Incorporating SBRG and MBRG in a Pairwise Model

We used the IBPRMF as the pairwise baseline model. As our focus is the positive-only feedback datasets in streaming scenarios, in which we have offline (batch) and online (streaming) updates, in this section, we present the IBPRMF adjustments to enable incorporating possible relevant items in both learning schemes. Our primary purpose is to combine negative and relevant items sampling to provide more accurate latent factor updates. Traditionally, IBPRMF has a triple (u, i, j) as input to the model training/update. On the other hand, to deal with negative and relevant items, we change the IBPRMF update rules, adding the instance prediction probability parameter p. Now, the IBPRMF model has as input (u, i, j, p), where u represents the current user in the instance, i is the current positive item, j is the negative item, and p parameter represents the probability for the $\widehat{r_{uij}}$ prediction. IBPRMF model receives as data the user's latent factor vectors A, item's latent factor vectors B, learning rate η, and regularization λ (Algorithm 8).

Algorithm 8 presents the IBPRMF update rules. In Lines 1 and 2, we predicted (u, i) and (u, j) pairs, respectively. Line 3 presents the instance prediction $\widehat{r_{uij}}$, which we generate by the difference between (u, i) and (u, j) predictions, multiplied by the instance probability p. Next, line 4 calculates the instance prediction error $\widehat{err}_{uij}$, which we used to update the user u (Line 5), item i (Line 6), and candidate item j (Line 7) latent factors vector.

Algorithm 8: BPRMF recommender - `model_update`.

Data: A: user's latent factor vectors, B: item's latent factor vectors, η: learning rate, λ: regularization rate.

Input: (u, i, j, p), where u: user identifier, i: positive item identifier, j: negative item identifier, p: probability value for (i, j) pair.

1 $\widehat{r}_{ui} \leftarrow A_u \cdot B_i$
2 $\widehat{r}_{uj} \leftarrow A_u \cdot B_j$
3 $\widehat{r}_{uij} \leftarrow (\widehat{r}_{ui} - \widehat{r}_{uj}) \times p$
4 $\widehat{err}_{uij} \leftarrow -\log(\sigma(\widehat{r}_{uij}))$
5 $\overrightarrow{A_u} \leftarrow \overrightarrow{A_u} + \eta \times \left(\widehat{err}_{uij} \times (B_i - B_j) - \lambda \times \overrightarrow{A_u}\right)$
6 $\overrightarrow{B_i} \leftarrow \overrightarrow{B_i} + \eta \times \left(\widehat{err}_{uij} \times (A_u) - \lambda \times \overrightarrow{B_i}\right)$
7 $\overrightarrow{B_j} \leftarrow \overrightarrow{B_j} + \eta \times \left(\widehat{err}_{uij} \times (-A_u) - \lambda \times \overrightarrow{B_j}\right)$

In Algorithm 9, we introduce the IBPRMF batch learning schema, which uses the dataset training samples (O_b^+), the available items I, the probabilities for positive-negative p_{ij}, positive-relevant p_{ir}, and relevant-negative p_{rj} pairs, the number of negative samples per instance k, and the number of training epochs. The IBPRMF batch training phase has three steps: updating the sampling structure, generating training data, and recommender model training.

Updating sampling structure: To use the provided sampling strategies SBRG and MBRG, we first must update the sampling structure using the training samples O_b^+ (Algorithm 9, Lines 1 to 3). For each instance (u, i), we call the `update_memory` function, which represents the SBRG (Algorithm 6) or MBRG (Algorithm 7) strategies.

Generating training data: Next, we have to create the training instances D (Algorithm 9, line 4) also considering each instance (u, i) in O_b^+ (Algorithm 9, line 5). However, we must perform the negative-relevant sampling k times (lines 6 and 7). We select 2 (two) negatives (j_1 and j_2, Algorithm 9 line 8) items from the available items I, except for those that the user interact with (I_u^+) or are in the set of relevant items I_u^r. We also select 2 (two) relevant (r_1 and r_2, Algorithm 9 line 9) items from I_u^r. We then use the selected negatives and relevant items combined with the current instance (u, i) to create three new training instances (C) (Algorithm 9, line 10). Each generated instance receives their own probability; for the positive-negative (i, j_1), positive-relevant (i, r_1), and relevant-negative (r_2, j_2) pairs, we use the p_{ij}, p_{ir}, and p_{rj} probabilities, respectively. Finally, we appended the generated instances C in the training dataset D (Algorithm 9, line 11).

Recommender Model Training: We perform the recommender model training z times (Algorithm 9, lines 15 and 16), where z represents the number of training epochs. As we perform the recommender model batch training, we shuffle the training instances D before each training step (Algorithm 9, line 17). Next, for each generated (u, i, j, p) instance (Algorithm 9, line 18), we call the `model_update` function (Algorithm 9, line 19) to use the IBPRMF update rules, introduce in ALgorithm 8.

Algorithm 10 presents the streaming processing phase of the IBPRMF recommender model, which we adapted to incorporate the provided sampling

Algorithm 9: Batch processing phase of the IBPRMF model.

Data: O_b^+: batch dataset instances, I: available items set, p_{ij}: probability for positive-negative pair, p_{ir}: probability for positive-relevant pair, p_{rj}: probability for relevant-negative pair, k: number of negative samples per instance, z: number of training epochs.

1 **foreach** $(u, i) \in O_b^+$ **do**
2 | `update_memory(u,i)` ▷ Updating user relevant items set (Algorithms 6 and 7)
3 **end**
4 $D \leftarrow \emptyset$
5 **foreach** $(u, i) \in O_b^+$ **do**
6 | $x \leftarrow 0$
7 | **while** $x < k$ **do**
8 | | draw j_1 and j_2 from $I \setminus (I_u^+ \cup I_u^r)$, such that $j_1 \neq j_2$ ▷ Sampling negatives items.
9 | | draw r_1 and r_2 from I_u^r, such that $r_1 \neq r_2$ ▷ Sampling relevant items.
10 | | $C \leftarrow \{(u, i, j_1, p_{ij}), (u, i, r_1, p_{ir}), (u, r_2, j_2, p_{rj})\}$
11 | | $D \leftarrow D \cup C$
12 | | $x \leftarrow x + 1$
13 | **end**
14 **end**
15 $x \leftarrow 0$
16 **while** $x < z$ **do**
17 | `shuffle`(D)
18 | **foreach** $(u, i, j, p) \in D$ **do**
19 | | `model_update`(u, i, j, p) ▷ Adjusted IBPRMF model update rules (Algorithm 8)
20 | **end**
21 | $x \leftarrow x + 1$
22 **end**

strategies SBRG and MBRG. In the streaming-incremental model training, we use the incoming dataset instance O_s^+, the available items I, and the prediction probabilities p_{ij}, p_{ir}, and p_{rj}. Naturally, we must perform a single pass-through data in the streaming process. In this sense, we also perform the negatives and relevant sampling a single time. For each emerged instance (u, i) in the stream (Algorithm 10, line 1), we select 2 (two) negatives items (j_1 and j_2, Algorithm 10 line 2) from the set of available items I that the user has not interacted with, or that are not possibly relevant. Additionally, we also sampled 2 (two) relevant

(r_1 and r_2, Algorithm 10 line 3) items from I_u^r. Next, we call the `model_update` function (Algorithm 8) to update the IBPRMF model by combining the positive-negative (line 4), positive-relevant (line 5) and relevant-negative pairs (line 6). Finally, we must update the relevant strategy (SBRG or MBRG) structure calling the `update_memory` function (Algorithm 10, line 7).

Algorithm 10: Streaming processing phase of the IBPRMF model.

Data: O_s^+: dataset incoming instances, I: available items set, p_{ij}: probability for positive-negative pair, p_{ir}: probability for positive-relevant pair, p_{rj}: probability for relevant-negative pair.

1 **foreach** $(u, i) \in O_s^+$ **do**
2 draw j_1 and j_2 from $I \setminus (I_u^+ \cup I_u^r)$, such that $j_1 \neq j_2$ ▷ Sampling negative items.
3 draw r_1 and r_2 from I_u^r, such that $r_1 \neq r_2$ ▷ Sampling relevant items.
4 `model_update`(u, i, j_1, p_{ij}) ▷ Algorithm 8
5 `model_update`(u, i, r_1, p_{ir}) ▷ Algorithm 8
6 `model_update`(u, r_2, j_2, p_{rj}) ▷ Algorithm 8
7 `update_memory(u,i)` ▷ Updating user relevant items set (Algorithms 6 and 7)
8 **end**

7.4.2 Incorporating SBRG and MBRG in a Point-wise Model

This section presents the combination of the relevant generation strategies (SBRG and MBRG) with the PMF model, previously introduced in Chapter 3. PMF is a point-wise model, so we must consider this characteristic when sampling negative and relevant items.

Algorithm 11 presents the PMF recommender model adjusted update rules to consider the prediction probability term p in each instance. Algorithm 11 uses the user's latent factor matrix A, the item's latent factor matrix B, the learning rate η, and the regularization rate λ. As input, PMF receives the user u, item i, the item score $s \in \{0, 1\}$, and the prediction probability p. First, we must recover the user-item (u, i) prediction rating r_{ui} applying the dot product in their latent factor vectors A_u and B_i, respectively (Algorithm 11, line 1). As we are working with the probabilistic matrix factorization, we obtained the probabilistic prediction $\widehat{p}_{ui}$ using the sigmoid function σ in the predicted rating r_{ui}, multiplied by the instance prediction probability p (Algorithm 11, line 2).

Next, we use the binary cross entropy loss to calculate the prediction error $\widehat{err}_{ui}$ for instance (Algorithm 11, line 3). Finally, we use the prediction error to update the user A_u and item B_i latent factor vectors (Algorithm 11, lines 4 and 5).

Algorithm 11: PMF Recommender - `model_update`.

Data: A: item's latent factor vectors, B: item's latent factor vectors, η: learning rate, λ: regularization rate.

Input: (u, i, s, p), where u: user, i: item, s: score $\in \{0, 1\}$, p: probability value for item (i).

1 $\widehat{r}_{ui} \leftarrow A_u \cdot B_i$
2 $\widehat{p}_{ui} \leftarrow \sigma(\widehat{r}_{ui} \times p)$
3 $\widehat{err}_{ui} \leftarrow -r \times \log(\widehat{p}_{ui} + 1^{-10}) + (1.0 - r) \times \log(1 - \widehat{p}_{ui} + 1^{-10})$
4 $\overrightarrow{A_u} \leftarrow \overrightarrow{A_u} + \eta \times \left(\widehat{err}_{ui} \times B_i - \lambda \times \overrightarrow{A_u}\right)$
5 $\overrightarrow{B_i} \leftarrow \overrightarrow{B_i} + \eta \times \left(\widehat{err}_{ui} \times A_u - \lambda \times \overrightarrow{B_i}\right)$

In algorithms 12 and 13, we present the batch and the streaming learning schemes of the PMF model, respectively. In Algorithm 12, we introduce the PMF batch learning schema, which uses the dataset training samples (O_b^+), the available items I, the probabilities for positive p_i, negative p_j, and relevant p_r items, the number of negative samples per instance k, and the number of training epochs. Similarly to the IBPRMF model, the PMF batch learning phase has three steps: updating the sampling structure, generating training data, and recommender model training.

The first step, **update sampling structure**, is the same as for the IBPRMF model (Algorithm 12, lines 1 to 3). On the other hand, as the PMF is a point-wise model, to **generate the training data** (second step, Algorithm 12, lines 4 to 15), we analyze the positive, negative, and relevant items separately. For each user-item pair in the training data (Algorithm 12, line 5), we first add the positive instance in the training data D, which has the score equal to 1, and receives the p_i probability (Algorithm 12, line 6). Next, we must sample the negative and relevant items k times (Lines 7 and 8). The point-wise model does not combine positive and negative examples. In this sense, we must select only one negative (line 9) and one relevant (line 10) item. We then create two new instances C to represent the user-negative and user-relevant pairs, where both receive the score s equal zero (0) and the probabilities p_j and p_r, respectively. Finally, the new instances are appended to the training data D.

The third step, **recommender model training** (Algorithm 12, lines 16 to 23), performs the PMF model training z times, where z represents the number of epochs. In each step, we shuffle the training instances D, and then we use all instances to update the PMF model, using function `model_update`, which refers to the PMF update rules (Algorithm 11).

Algorithm 12: Batch processing phase of PMF model.

Data: O_b^+: dataset batch instances, p_i: probability for positive item, p_j: probability for negative item, p_j: probability for relevant item, k: Number of samples per positive instance, z: number of training epochs.

1 **foreach** $(u, i) \in O_b^+$ **do**
2 | `update_memory(u,i)` ▷ Updating user relevant items set (Algorithms 6 and 7)
3 **end**
4 $D \leftarrow \emptyset$
5 **foreach** $(u, i) \in O_b^+$ **do**
6 | $D \leftarrow D \cup (u, i, 1, p_i)$
7 | $x \leftarrow 0$
8 | **while** $x < k$ **do**
9 | | draw j from $I \setminus (I_u^+ \cup I_u^r)$ ▷ Sampling negative item.
10 | | draw r from I_u^r ▷ Sampling relevant item.
11 | | $C \leftarrow \{(u, j, 0, p_j), (u, r, 0, p_r)\}$
12 | | $D \leftarrow D \cup C$
13 | | $x \leftarrow x + 1$
14 | **end**
15 **end**
16 $x \leftarrow 0$
17 **while** $x < z$ **do**
18 | `shuffle`(D)
19 | **foreach** $(u, i, s, p) \in D$ **do**
20 | | `model_update`(u, i, s, p) ▷ Adjusted PMF model update rules (Algorithm 11)
21 | **end**
22 | $x \leftarrow x + 1$
23 **end**

Algorithm 13 presents the adaptations in the PMF streaming processing phase to incorporate the provided sampling strategies SBRG and MBRG. In the streaming-incremental model training, we use the incoming dataset instance O_s^+, the available items I, and the prediction probabilities p_i, p_j, and p_r. We

perform the negative and relevant sampling a single time, passing a single time through the data. For each emerged instance (u, i) in the stream (Algorithm 13, line 1), we first update the PMF model with the incoming instance (Algorithm 13 line 2). Then we select one negative item (j, Algorithm 13, line 3) from the set of available items I that the user has not interacted with or that are not possibly relevant and update the PMF model with the $(u, j, 0, p_j)$ parameters (Algorithm 13, line 4). Additionally, we also sampled one relevant (r, Algorithm 13 line 5) item from I_u^r. Next, we call the `model_update` function (Algorithm 11) to update the IBPRMF model with the $(u, r, 0, p_r)$ parameters (Algorithm 13, line 6). Finally, we must update the relevant strategy (SBRG or MBRG) structure calling the `update_memory` function (Algorithm 13, line 7).

Algorithm 13: Streaming processing phase of PMF model.

Data: O_s^+: dataset incoming instances, A: user's latent factor vectors, B: item's latent factor vectors, p_i: probability for positive item, p_j: probability for negative item, p_j: probability for relevant item

1 **foreach** $(u, i) \in O_s^+$ **do**
2 `model_update`$(u, i, 1, p_i)$ ▷ Algorithm 11
3 draw j from $I \setminus (I_u^+ \cup I_u^r)$ ▷ Get j from the unknown and not relevant items set.
4 `model_update`$(u, j, 0, p_j)$ ▷ Algorithm 11
5 draw r from I_u^r ▷ Get r from the relevant items set.
6 `model_update`$(u, r, 0, p_r)$ ▷ Algorithm 11
7 `update_memory(u,i)` ▷ Updating user relevant items set (Algorithms 6 and 7)
8 **end**

7.5 Experimental Setup

We compare our proposed MBRG and SBRG strategies with and against the Uniform Random Sampling algorithm previously introduced in Section 7.2. We incorporate the sampling strategies in the adjusted PMF (SALAKHUTDINOV; MNIH, 2007) and IBPRMF (RENDLE et al., 2012) recommender models as baselines.

We tested the following hyper-parameter values in the IBPRMF and PMF incremental recommender models: learning rate $\in$ {0.01, 0.001}, regularization rate equal to 0.01, latent factors $\in$ {32, 64, 128}, number of negatives items in

training phase ∈ {1, 3, 5, 10}, batch training epochs equal to 10. We also tested different values for the p parameter in the BPRMF (pairwise) and PMF (pointwise) methods, where $p \in$ {(0.6,0.2,0.2), (0.2,0.6,0.2), (0.2,0.2,0.6)}, which are the probability values for (p_{ij}, p_{jr}, p_{rj}) in BPRMF model and the (p_i, p_j, p_r) in PMF model.

We choose the best hyper-parameters based on a grid search, which exhaustively generates candidates from the specified grid of parameter values. For each combination, we use the entire dataset. In this sense, the parameter tuning is not part of the method's processing time. We replicate each best experimental setting ten times in this work, so the results depict the average and standard deviation of recall values. We selected a single random seed for each replication and used it with all optimizers to enable further paired comparisons and the application of the statistical significance test.

Finally, we incorporate hypothesis testing to determine whether one sampling strategy outperforms others significantly, as presented in Chapter 4. We execute each experiment ten times for finding any statistically significant difference.

7.6 Results and Analysis

Tables 7.1 and 7.2 present the obtained results of the IBPRMF and PMF models. We mark the best result for the dataset and the number of negative items in bold. Considering the IBPRMF model (Table 7.1), we can observe that our sampling strategies outperformed the traditional IBPRMF recommender model in most cases. The Amazon Books dataset results obtained the best RECALL@10 value with the IBPRMF+SBRG strategy with five negative items, increasing by up to 1.9 percentage points (from 0.01 to 0.029). For the Movie Lens 1M dataset, we obtained an increase by up to 6.9 percentage points (from 0.115 to 0.184) in RECALL@10 values, with the IBPRMF+MBRG strategy and ten negative-relevant samples. Concerning the Movie Tweetings dataset, the IBPRMF+MBRG strategy increases the RECALL@10 values by up to 15.8 percentage points (from 0.065 to 0.223), with a single negative-relevant sample per instance. Considering the SMDI-200UE dataset, the IBPRMF+MBRG strategy enables an increase of 6.8 percentage points (from 0.218 to 0.286). Finally, considering the TaFeng dataset, we obtained an increase of 4.2 percentage points (from 0.120 to 0.162) in RECALL@10 values with the IBPRMF+SBRG method.

Table 7.1: RECALL@10 and NDCG@10 values obtained by the IBPRMF, IBPRMF+MBRG, and IBPRMF+SBRG models in the tested datasets.

D	# N	BPRMF		BPRMF+MBRG		BPRMF+SBRG	
		RECALL@10	NDCG@10	RECALL@10	NDCG@10	RECALL@10	NDCG@10
Amazon Books	1	0.010 ± 0.0002	0.004 ± 0.0001	0.011 ± 0.0002	0.005 ± 0.0001	0.010 ± 0.0002	0.005 ± 0.0001
	3	0.009 ± 0.0001	0.004 ± 0.0001	0.020 ± 0.0006	0.010 ± 0.0003	0.027 ± 0.0005	0.014 ± 0.0003
	5	0.009 ± 0.0002	0.004 ± 0.0001	0.020 ± 0.0005	0.010 ± 0.0003	**0.029 ± 0.0006**	0.015 ± 0.0003
	10	0.009 ± 0.0001	0.005 ± 0.0001	0.018 ± 0.0002	0.009 ± 0.0001	0.026 ± 0.0003	0.013 ± 0.0002
Movie Lens 1M	1	0.110 ± 0.0009	0.054 ± 0.0004	0.143 ± 0.0012	0.072 ± 0.0006	0.105 ± 0.0016	0.052 ± 0.0007
	3	0.116 ± 0.0005	0.057 ± 0.0002	0.173 ± 0.0008	0.086 ± 0.0003	0.140 ± 0.0010	0.068 ± 0.0005
	5	0.116 ± 0.0007	0.057 ± 0.0005	0.180 ± 0.0007	0.090 ± 0.0004	0.148 ± 0.0004	0.072 ± 0.0002
	10	0.115 ± 0.0006	0.056 ± 0.0003	**0.184 ± 0.0005**	**0.091 ± 0.0003**	0.152 ± 0.0007	0.074 ± 0.0003
Movie Tweetings	1	0.015 ± 0.0023	0.007 ± 0.0012	**0.223 ± 0.0021**	0.124 ± 0.0011	0.218 ± 0.0020	0.107 ± 0.0019
	3	0.065 ± 0.0010	0.029 ± 0.0004	0.191 ± 0.0015	0.116 ± 0.0007	0.166 ± 0.0012	0.080 ± 0.0010
	5	0.059 ± 0.0004	0.027 ± 0.0002	0.200 ± 0.0014	0.122 ± 0.0008	0.168 ± 0.0011	0.084 ± 0.0005
	10	0.054 ± 0.0006	0.025 ± 0.0003	**0.222 ± 0.0009**	**0.136 ± 0.0007**	0.176 ± 0.0010	0.089 ± 0.0007
SMDI-200UE	1	0.015 ± 0.0015	0.007 ± 0.0008	0.254 ± 0.0060	0.156 ± 0.0032	0.249 ± 0.0066	0.153 ± 0.0035
	3	0.218 ± 0.0026	0.129 ± 0.0012	**0.286 ± 0.0005**	**0.174 ± 0.0005**	0.258 ± 0.0011	0.159 ± 0.0005
	5	0.232 ± 0.0014	0.133 ± 0.0009	0.268 ± 0.0006	0.163 ± 0.0004	0.231 ± 0.0012	0.141 ± 0.0008
	10	0.207 ± 0.0013	0.118 ± 0.0009	0.238 ± 0.0009	0.142 ± 0.0006	0.188 ± 0.0013	0.107 ± 0.0008
TaFeng	1	0.010 ± 0.0002	0.004 ± 0.0001	0.038 ± 0.0052	0.023 ± 0.0035	0.036 ± 0.0053	0.022 ± 0.0037
	3	0.057 ± 0.0059	0.034 ± 0.0034	**0.155 ± 0.0006**	**0.091 ± 0.0004**	**0.162 ± 0.0007**	**0.087 ± 0.0004**
	5	0.119 ± 0.0020	0.064 ± 0.0011	0.144 ± 0.0006	0.084 ± 0.0004	0.151 ± 0.0005	0.082 ± 0.0003
	10	0.120 ± 0.0007	0.064 ± 0.0005	0.128 ± 0.0004	0.073 ± 0.0003	0.128 ± 0.0006	0.067 ± 0.0003

Considering the PMF model results (Table 7.1), we can also observe that our negative-relevant strategies provided superior performance in most cases, compared with the traditional uniform random sampling of negative items. The PMF+MBRG strategy increased by 7.4 percentage points (from 0.024 to 0.098) in the RECALL@10 values for the Amazon Books dataset. Considering the Movie Lens 1M dataset, we also obtained the best result with the PMF+MBRG strategy, which enables an increase of 0.8 percentage points. For the Movie Tweentings dataset, the PMF+MBRG strategy increased RECALL@10 values by up to 17.4 percentage points (from 0.126 to 0.300). Considering the SMDI-200UE dataset, the best result represents an increase of 0.6 percentage points obtained by the PMF+MBRG strategy. Finally, considering the TaFeng dataset, the PMF+SBRG strategy provided de best result, with an increase of 13 percentage points (from 0.046 to 0.176) in the RECALL@10 values.

Concerning the IBPRMF and PMF best results in each dataset, in the Amazon Books, the PMF+MBRG strategy provided superior performance than the IBPRMF-SBRG, with an increase of 6.9 percentage points (from 0.029 to 0.098). In contrast, the IBPRMF+MBRG outperforms the PMF+MBRG by up to 5.9 percentage points (from 12.5 to 18.4) in the Movie Lens 1M dataset. Considering the Movie Tweetings dataset, we obtained the best result by the PMF+MBRG, outperforming the IBPRMF+MBRG by up to 7.7 percentage points (from 22.3 to

Table 7.2: RECALL@10 and NDCG@10 values obtained by the PMF, PMF+MBRG, and PMF+SBRG models in the tested datasets.

D	# N	PMF		PMF+MBRG		PMF+SBRG	
		RECALL@10	NDCG@10	RECALL@10	NDCG@10	RECALL@10	NDCG@10
Amazon Books	1	0.010 ± 0.0002	0.004 ± 0.0001	0.015 ± 0.0002	0.007 ± 0.0001	0.011 ± 0.0002	0.005 ± 0.0001
	3	0.010 ± 0.0002	0.004 ± 0.0001	0.023 ± 0.0012	0.011 ± 0.0006	0.023 ± 0.0007	0.011 ± 0.0004
	5	0.010 ± 0.0003	0.005 ± 0.0001	0.075 ± 0.0010	0.040 ± 0.0007	0.047 ± 0.0007	0.022 ± 0.0005
	10	0.024 ± 0.0005	0.011 ± 0.0003	**0.098 ± 0.0007**	**0.053 ± 0.0005**	0.064 ± 0.0005	0.031 ± 0.0003
Movie Lens 1M	1	0.111 ± 0.0013	0.054 ± 0.0007	0.121 ± 0.0008	0.060 ± 0.0005	0.077 ± 0.0017	0.038 ± 0.0010
	3	0.114 ± 0.0005	0.056 ± 0.0002	**0.125 ± 0.0005**	**0.063 ± 0.0004**	0.082 ± 0.0007	0.041 ± 0.0005
	5	0.114 ± 0.0007	0.056 ± 0.0004	0.123 ± 0.0008	0.062 ± 0.0004	0.082 ± 0.0007	0.041 ± 0.0003
	10	0.117 ± 0.0007	0.057 ± 0.0003	0.122 ± 0.0013	0.062 ± 0.0005	0.084 ± 0.0010	0.042 ± 0.0006
Movie Tweetings	1	0.015 ± 0.0019	0.007 ± 0.0010	0.024 ± 0.0032	0.012 ± 0.0017	0.029 ± 0.0048	0.014 ± 0.0025
	3	0.018 ± 0.0027	0.009 ± 0.0014	0.064 ± 0.0048	0.032 ± 0.0023	0.158 ± 0.0033	0.077 ± 0.0015
	5	0.031 ± 0.0033	0.015 ± 0.0015	0.174 ± 0.0060	0.097 ± 0.0045	0.225 ± 0.0021	0.122 ± 0.0016
	10	0.126 ± 0.0020	0.057 ± 0.0009	**0.300 ± 0.0010**	**0.210 ± 0.0009**	0.278 ± 0.0016	0.161 ± 0.0008
SMDI-200UE	1	0.015 ± 0.0017	0.007 ± 0.0009	0.024 ± 0.0042	0.013 ± 0.0026	0.021 ± 0.0036	0.011 ± 0.0023
	3	0.023 ± 0.0043	0.012 ± 0.0027	0.144 ± 0.0175	0.095 ± 0.0114	0.158 ± 0.0063	0.103 ± 0.0039
	5	0.070 ± 0.0148	0.044 ± 0.0100	**0.228 ± 0.0012**	**0.148 ± 0.0008**	0.153 ± 0.0018	0.099 ± 0.0012
	10	0.224 ± 0.0026	0.136 ± 0.0013	0.211 ± 0.0006	0.139 ± 0.0006	0.136 ± 0.0018	0.083 ± 0.0014
TaFeng	1	0.010 ± 0.0002	0.004 ± 0.0001	0.011 ± 0.0002	0.005 ± 0.0001	0.011 ± 0.0003	0.005 ± 0.0002
	3	0.010 ± 0.0003	0.005 ± 0.0001	0.016 ± 0.0010	0.008 ± 0.0007	0.036 ± 0.0056	0.021 ± 0.0036
	5	0.011 ± 0.0003	0.005 ± 0.0002	0.059 ± 0.0038	0.036 ± 0.0024	0.154 ± 0.0032	0.087 ± 0.0015
	10	0.046 ± 0.0068	0.028 ± 0.0042	0.121 ± 0.0004	0.073 ± 0.0003	**0.176 ± 0.0009**	**0.099 ± 0.0006**

30.0). The IBPRMF+MBRG also provided the best results in the SMDI-200UE dataset, increasing RECALL@10 values by up to 5.8 percentage points (22.8 to 28.6). On the other hand, considering the TaFeng dataset, PMF-SBRG obtained the best result of the experiments, with an increase of 1.4 percentage points (from 16.2 to 17.6).

We can justify the obtained results in the analyzed datasets by comparing their number of instances, users, and items. In the small datasets, Movie Lens and SMDI-200UE, the IBPRMF provided superior performance. Nonetheless, the PMF recommender obtained the best results considering the Amazon Books, Movie Tweetings, and TaFeng datasets. We can also observe that the higher differences between the provided negative-relevant strategies and the random sampling were obtained in the large datasets. In the Amazon Books, Movie Tweeting, and TaFeng datasets, our strategies increased RECALL@10 values by up to 7.4, 17.4, and 13.0 percentage points, respectively. On the other hand, for the Movie Lens and SMDI-200UE datasets, we obtained an increase of up to 6.9 and 5.8, respectively. In this sense, our proposed sampling strategies provided the best performance in datasets with a higher number of users and items, which is reasonable because when the higher the number of users and items, the smaller the number o instances in which each user and item appear. Furthermore, as the number of incoming users and items increases, the incidence

of cold-start also increases. In this sense, training and updating recommender models with negatives and relevant items (with specific probabilities) can be successfully used in cold-start scenarios.

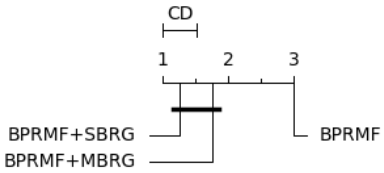
Figures 7.1 and 7.2 present the results of Friedman and Nemenyi's statistical significance tests, considering the experiments of BPRMF and PMF model's variants in each analyzed dataset. The legends present the Friedman p-values and show significant differences, assuming a 95% (p-value < 0.05) confidence level. We consider all the experiment results in the statistical significance analysis concerning the replications of each experiment setting (recommendation strategy, dataset, and number of samples per positive instance). Therefore, the graph of the critical distance obtained by the Nemenyi test shows each recommendation strategy's rankings and significant differences. We can observe that the behavior of the results is similar to that presented in Tables 7.1 and 7.2.

Considering the BPRMF model variants, we observe that the proposed negative-relevant sampling strategies, combined with the BPRMF model, outperform the traditional uniform random sampling and present statistically significant differences in most cases. In the Amazon Books dataset, the BPRMF-SBRG presents the best RECALL@10 and NDCG@10 results. However, there are no significant differences compared with the BPRMF-MBRG strategy. On the other hand, both strategies presented significant differences compared with the traditional BPRMF model.

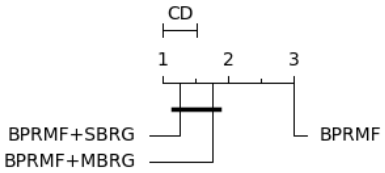
For the Movie Lens 1M dataset, the BPRMF-MBRG model obtained the best results compared with the BPRMF-SBRG and BPRMF strategies, showing a statistically significant difference with a 95% of confidence level. In contrast to what we observed in the Amazon Books results, for the Movie Lens 1M dataset, the BPRMF-SBRG and BPRMF strategies do not differ, considering both RECALL@10 and NDCG@10 values. We observed the same behavior in the SMDI-200UE dataset.

Considering the Movie Tweetings dataset, the BPRMF-MBRG strategy outperforms and differs from the other strategies. On the other hand, BPRMF-SBRG and BPRMF strategies do not present significant differences considering only the RECALL@10 values. When we look for the NDCG@10 values, the BPRMF-SBRG model also provides better results and differs from the traditional BPRMF model.

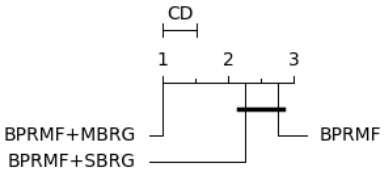
Finally, considering the TaFeng dataset, we observe a different behavior in the obtained results. Considering the RECALL@10 values, the BPRMF-SBRG strategy is in the first position and does not present a significant difference



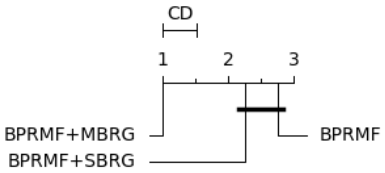
(a) Amazon Books RECALL@10 ($p = 7.68 \times 10^{-15}$).



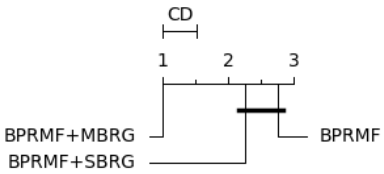
(b) Amazon Books NDCG@10 ($p = 4.31 \times 10^{-14}$).



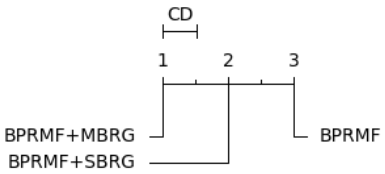
(c) Movie Lens RECALL@10 ($p = 7.68 \times 10^{-15}$).



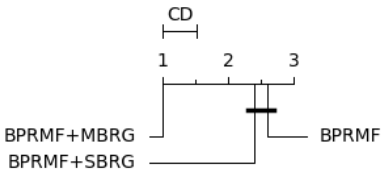
(d) Movie Lens NDCG@10 (p-value $= 7.68 \times 10^{-15}$).



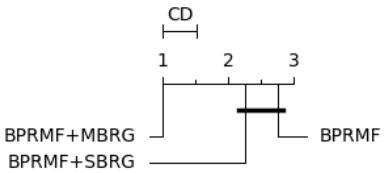
(e) Movie Tweetings RECALL@10 (p-value $= 4.24 \times 10^{-18}$).



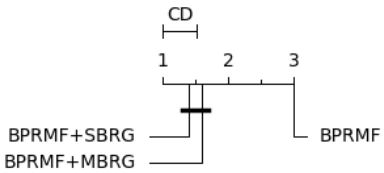
(f) Movie Tweetings NDCG@10 (p-value $= 4.24 \times 10^{-18}$).



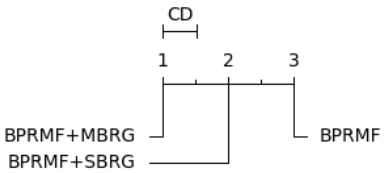
(g) SMDI-200UE RECALL@10 (p-value $= 1.42 \times 10^{-11}$).



(h) SMDI-200UE NDCG@10 (p-value $= 1.42 \times 10^{-11}$).



(i) TaFeng RECALL@10 (p-value $= 9.13 \times 10^{-14}$).

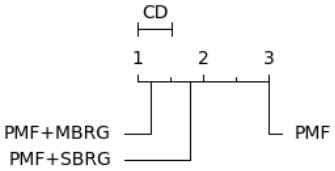


(j) TaFeng NDCG@10 (p-value $= 5.01 \times 10^{-14}$).

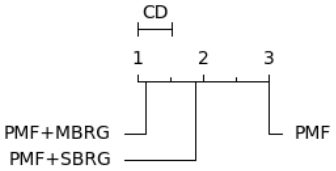
Figure 7.1: Critical distances of the Nemenyi test for tested datasets results obtained by the BPRMF model variants. All p-values refer to the Friedman test.

compared to the BPRMF-MBRG strategy. However, looking at the NDCG@10 values, BPRMF-MBRG outperformed others, and both presented significant differences. However, all the results show us that the proposed strategies obtained

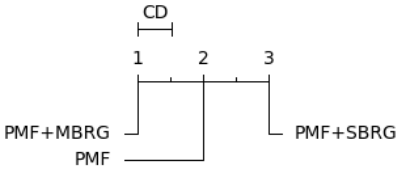
competitive results compared with the traditional BPRMF model. In this sense, combining the negative-relevant sampling in the pairwise model is an alternative to the uniform random sampling of negative items.



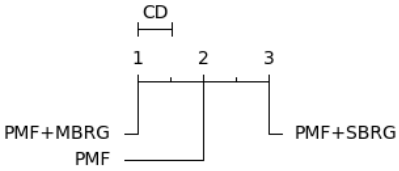
(a) Amazon Books RECALL@10 (p-value = 3.88×10^{-12}).



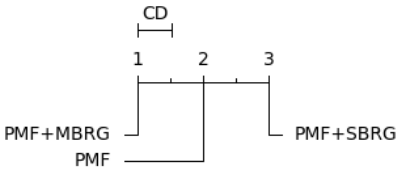
(b) Amazon Books NDCG@10 (p-value = 4.09×10^{-13}).



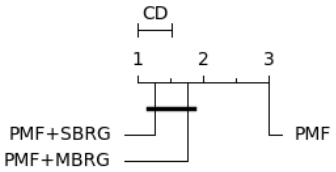
(c) Movie Lens RECALL@10 (p-value = 4.25×10^{-18}).



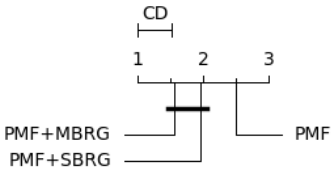
(d) Movie Lens NDCG@10 (p-value = 4.25×10^{-18}).



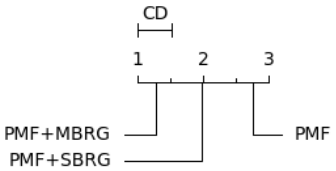
(e) Movie Tweetings RECALL@10 (p-value = 4.29×10^{-12}).



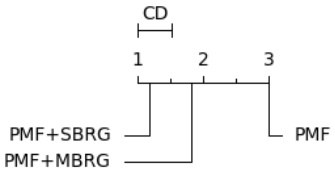
(f) Movie Tweetings NDCG@10 (p-value = 4.29×10^{-12}).



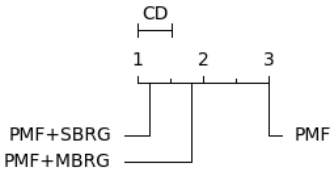
(g) SMDI-200UE RECALL@10 (p-value = 5.17×10^{-07}).



(h) SMDI-200UE NDCG@10 (p-value = 1.13×10^{-08}).



(i) TaFeng RECALL@10 (p-value = 2.07×10^{-10}).



(j) TaFeng NDCG@10 (p-value = 1.57×10^{-10}).

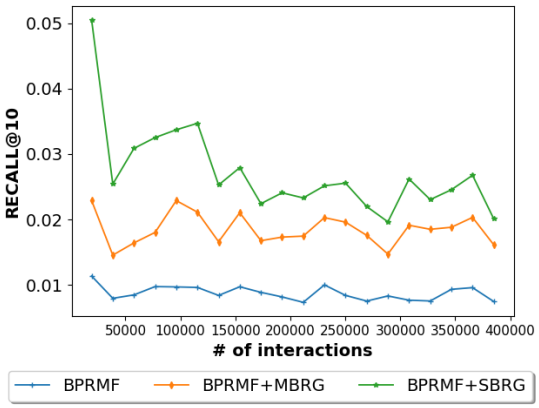
Figure 7.2: Critical distances of the Nemenyi test for tested datasets results obtained by the PMF model variants. All p-values refer to the Friedman test.

Figure 7.2 shows the statistically significant analysis of the PMF model variant's results. Similar to what we observed for the BPRMF variants, the proposed sampling strategies combined with the point-wise model also outperform the uniform random sampling of unobserved items in most cases. However, the traditional PMF strategy outperforms our proposed SBRG sampling strategy for the point-wise model, as we can observe in the Movie Lens and Movie Tweetings datasets results (also presented in Table 7.2). On the other hand, the PMF-MBRG provided the best performance with significant differences in the RECALL@10 values considering both the Amazon Books, Movie Lens, Movie Tweetings, and SMDI-200UE datasets. Concerning the NDCG@10 values, the PMF-SBRG provided the best results for the Movie Tweetings and SMDI-200UE. This behavior shows us that although the PMF-MBRG provided more TOP@K corrected ranked items, the PMF-SBRG strategy is capable of ranking the positive instances at the first positions.

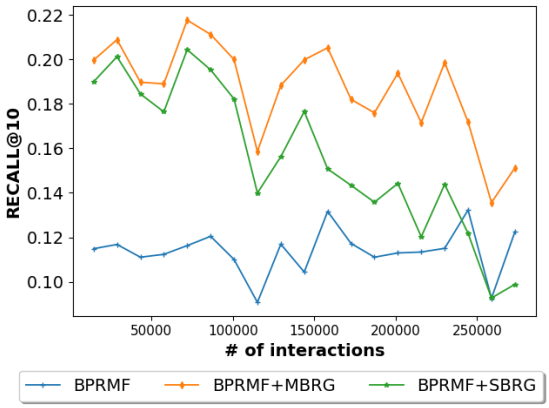
7.6.1 Streaming Analysis of the Results

This section presents the evolution of the RECALL@10 values according to new instances that emerged from the data stream. Figure **??** presents the windowed evaluation of RECALL@10 values by analyzing the best results considering each dataset and the number of selected samples. As this analysis generated an expressive number of experiment settings, the windowed evaluation only shows the best results for each dataset in the pairwise and point-wise models to plot the windowed evaluation. Thus, all the obtained results for the windowed assessment are presented in Appendix A. We considered a window with a size of 5% of the number of test interactions. Considering the Amazon Books dataset results (Figure 7.3a), we observe that the BPRMF-SBRG outperforms other BPRMF variants during the entire streaming test set. On the other hand, considering the Movie Lens 1M dataset, we obtained the best performance in the streaming analysis by the BPRMF-SBRG strategy. For the Movie Tweetings dataset, we can observe that both the proposed strategies, BPRMF-SBRG and BPRMF-SBRG, presented similar behavior in the plot evolution of RECALL@10 values with a visible best performance than the traditional BPRMF. Similar to what we obtained in the basic evaluation process (Table 7.1), the windowed evaluation of the SMDI-200UE dataset results presented the lowest increases than the other datasets. However, the proposed strategies also outperformed

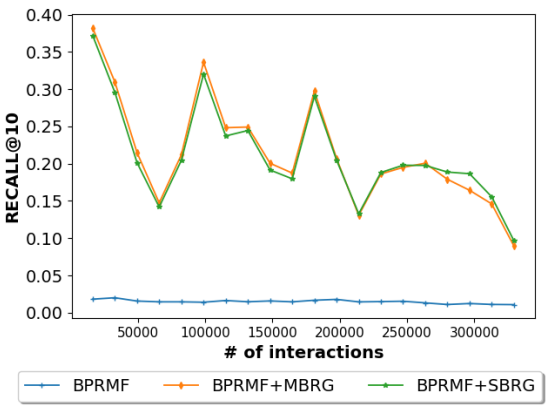
the BPRMF baseline. Finally, for the TaFeng dataset, we can observe that the proposed sampling strategies also performed better than the random sampling baseline in the entire instances streams.



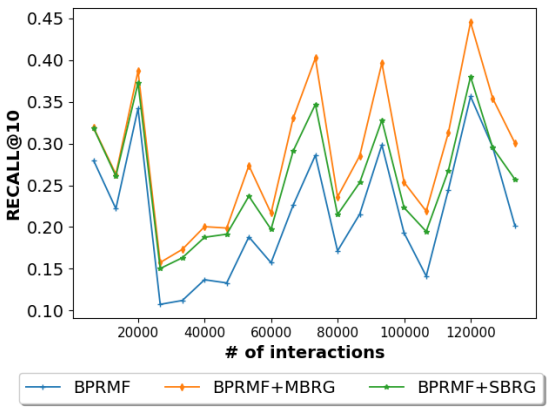
(a) Amazon Books - 5 samples



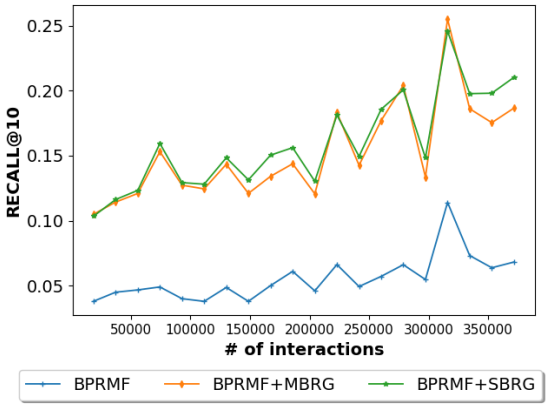
(b) Movie Lens - 10 samples



(c) Movie Tweetings - 1 sample

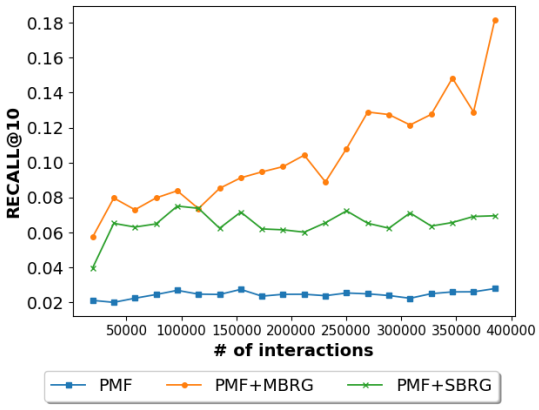


(d) SMDI-200UE - 3 sample

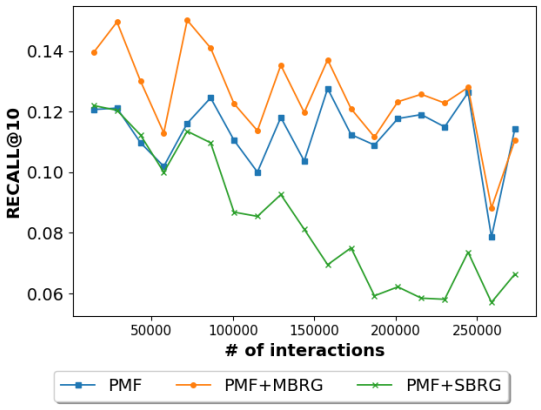


(e) TaFeng - 3 samples

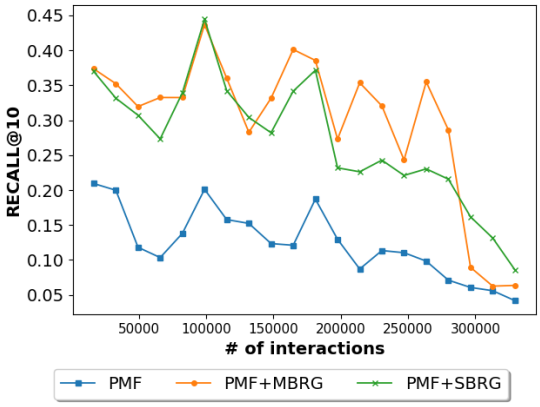
Figure 7.3: Windowed Evaluation of RECALL@10 values for the experiments of the BPRMF model variants.



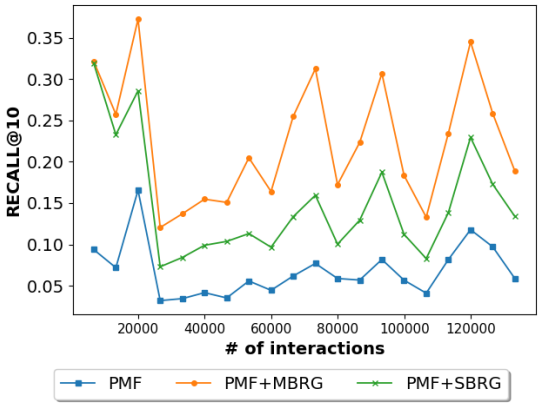
(a) Amazon Books - 10 samples



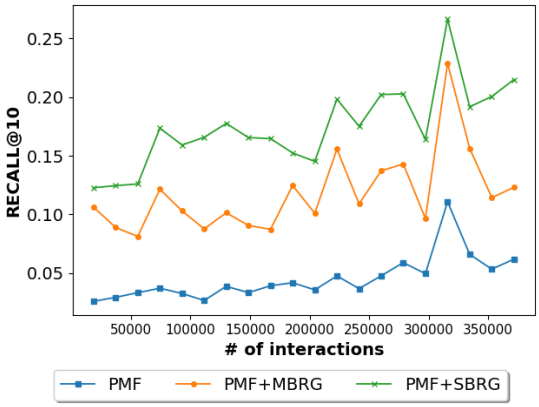
(b) Movie Lens - 3 samples



(c) Movie Tweetings - 10 sample



(d) SMDI-200UE - 5 sample



(e) TaFeng - 10 samples

Figure 7.4: Windowed Evaluation of RECALL@10 values for the experiments of the PMF model variants.

Figure 7.4 shows the windowed RECALL@10 values obtained using the PMF model variants in all the analyzed datasets. As we can observe in the presented plot evolution of RECALL@10 values, the PMF-MBRG presented the best results

in most of the tested datasets. The PMF-MBRG strategy outperforms other PMF variants in the Amazon Books, Movie Lens 1M, Movie Tweetings, and SMDI-200UE datasets. On the other hand, the PMF-SBRG strategy presented the best results in the TaFeng dataset.

Considering the Movie Lens dataset results, in which we also obtained the best performance with the PMF-MBRG strategy, we can observe similar behavior in the windowed evaluation and basic evaluation process (presented in Table 7.2), where the traditional PMF model outperformed the PMF-SBRG strategy. Considering that the similarity-based model analyzes the similarities between the user's positive and unobserved items for the Movie Lens 1M dataset, which has fewer users and items, the PMF-SBRG does not generate relevant items successfully. On the other hand, the similarity-based strategy performed better in datasets with large numbers of users and items. Considering that most of the available datasets have a lot of users and items, we can combine both MBRG and SBRG strategies with pairwise and point-wise models to obtain more accurate results.

7.7 Final Considerations

This chapter provided an analysis of negative sampling in OCCF scenarios. The presented recommendation strategies are variants of the traditional pairwise and point-wise models BPRMF and PMF, respectively. BPRMF and PMF need unobserved items to serve as negatives during model training and update. This study proposes two alternatives for the traditional uniform random sampling that were successfully combined with pairwise and point-wise methods. As observed in this chapter, the negative-relevant sampling strategies outperformed the tested negative sampling baseline in most of the presented results. The proposed methods increased the RECALL@10 values by up to 17.4 percentage points compared with the uniform sampling baseline.

8

Conclusion

This thesis focused on optimizing recommender models by providing alternatives for the latent factors' updates and negative sampling in OCCF streaming scenarios. In this work, the use of batch and incremental recommender models were analyzed in different aspects and scenarios, and the objectives were aligned with the developed analysis. Considering the availability of new retail datasets for OCCF with concept drift and cold-start incidence, this work provides a novel supermarket data collection from four months of purchases in a physical supermarket store. A robust comparison protocol between batch and streaming recommender models was also introduced. Considering the provided dataset (Chapter 5), it was concluded that in real-world supermarket data, the streaming recommender models induced better recall rates than the batch learning approaches due to the incidence of cold-start and concept drifts. However, as the dataset was collected over four months, and although it will be helpful in the research community, I have as future work the acquisition of at least one year of supermarket transactions with more than 10 million user-item interactions.

Considering the analysis and implementation of adaptive recommender models, Chapter 6 presents five variants of four well know adaptive optimizers, i.e., Adam, AMSGrad, Nadam, and RMSprop, and provides results that show us the efficiency of adaptive learning rate optimizer combined with traditional and incremental MF model. I plan to investigate other adaptive learning rate optimizers in future work to analyze their incremental efficiency in updating MF models. I also intend to incorporate our optimizer variants into other MF recommender models, such as the PMF and BPRMF models presented in this

study. Furthermore, we plan to investigate the application of drift detectors as part of the learning process to adapt the model parameters according to changes in the data.

Focusing on the negative-relevant items sampling, presented in Chapter 7, this works studies the proper selection of unknown and possible relevant items to be used in the recommender models training. The proposed sampling strategies were combined in both pairwise and point-wise models. The proposed strategies enabled the recommender models to learn more efficient user and item features and then recommend more accurate items to users. In future works, I want to implement other sampling alternatives applied in streaming scenarios for comparison. Additionally, I plan to use other similarity measures for finding relevant items.

Furthermore, I envision combining the MF models with temporal dynamics since the supermarket/e-commerce scenarios have a large time dependency due to the specific behaviors, with a high incidence of cold-start and concept drifts.

References

ABADI, M.; BARHAM, P.; CHEN, J.; CHEN, Z.; DAVIS, A.; DEAN, J.; DEVIN, M.; GHEMAWAT, S.; IRVING, G.; ISARD, M.; KUDLUR, M.; LEVENBERG, J.; MONGA, R.; MOORE, S.; MURRAY, D. G.; STEINER, B.; TUCKER, P. A.; VASUDEVAN, V.; WARDEN, P.; WICKE, M.; YU, Y.; ZHENG, X. Tensorflow: A system for large-scale machine learning. In: KEETON, K.; ROSCOE, T. (Ed.). *12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016, Savannah, GA, USA, November 2-4, 2016*. USENIX Association, 2016. p. 265–283. Disponível em: <https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi>.

AGGARWAL, C. C. (Ed.). *Data Streams - Models and Algorithms*. Springer, 2007. v. 31. (Advances in Database Systems, v. 31). ISBN 978-0-387-28759-1. Disponível em: <https://doi.org/10.1007/978-0-387-47534-9>.

AL-GHOSSEIN, M.; MURENA, P.; ABDESSALEM, T.; BARRÉ, A.; CORNUÉJOLS, A. Adaptive collaborative topic modeling for online recommendation. In: PERA, S.; EKSTRAND, M. D.; AMATRIAIN, X.; O'DONOVAN, J. (Ed.). *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018*. ACM, 2018. p. 338–346. Disponível em: <https://doi.org/10.1145/3240323.3240363>.

BABÜROGLU, E. S.; DURMUSOGLU, A.; DERELI, T. Novel hybrid pair recommendations based on a large-scale comparative study of concept drift detection. *Expert Syst. Appl.*, v. 163, p. 113786, 2021. Disponível em: <https://doi.org/10.1016/j.eswa.2020.113786>.

BAI, T.; DU, P.; ZHAO, W. X.; WEN, J.; NIE, J. A long-short demands-aware model for next-item recommendation. *CoRR*, abs/1903.00066, 2019. Disponível em: <http://arxiv.org/abs/1903.00066>.

BALAKRISHNAN, J.; CHENG, C. H.; WONG, K.; WOO, K. Product recommendation algorithms in the age of omnichannel retailing - an intuitive clustering approach. *Comput. Ind. Eng.*, v. 115, p. 459–470, 2018. Disponível em: <https://doi.org/10.1016/j.cie.2017.12.005>.

BEEL, J.; GIPP, B.; LANGER, S.; BREITINGER, C. Research-paper recommender systems: a literature survey. *Int. J. on Digital Libraries*, v. 17, n. 4, p. 305–338, 2016. Disponível em: <https://doi.org/10.1007/s00799-015-0156-0>.

BI, Y.; SONG, L.; YAO, M.; WU, Z.; WANG, J.; XIAO, J. DCDIR: A deep cross-domain recommendation system for cold start users in insurance domain. In: *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*. ACM, 2020. p. 1661–1664. Disponível em: <https://doi.org/10.1145/3397271.3401193>.

BISONG, E. *Building machine learning and deep learning models on Google cloud platform*. [S.l.]: Springer, 2019.

BOBADILLA, J.; ORTEGA, F.; HERNANDO, A.; GUTIÉRREZ, A. Recommender systems survey. *Knowledge-based systems*, Elsevier, v. 46, p. 109–132, 2013.

BREIMAN, L. Bagging predictors. *Mach. Learn.*, v. 24, n. 2, p. 123–140, 1996. Disponível em: <https://doi.org/10.1007/BF00058655>.

BURLUTSKIY, N.; PETRIDIS, M.; FISH, A.; CHERNOV, A.; ALI, N. An investigation on online versus batch learning in predicting user behaviour. In: *Research and Development in Intelligent Systems XXXIII - Incorporating Applications and Innovations in Intelligent Systems XXIV. Proceedings of AI-2016, The Thirty-Sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, UK, December 13-15, 2016*. Springer, 2016. p. 135–149. Disponível em: <https://doi.org/10.1007/978-3-319-47175-4_9>.

ÇANO, E.; MORISIO, M. Characterization of public datasets for recommender systems. In: IEEE. *2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*. [S.l.], 2015. p. 249–257.

CHAE, D.; LEE, S.; LEE, S.; KIM, S. On identifying k-nearest neighbors in neighborhood models for efficient and effective collaborative filtering. *Neurocomputing*, v. 278, p. 134–143, 2018. Disponível em: <https://doi.org/10.1016/j.neucom.2017.06.081>.

CHANDRAMOULI, B.; LEVANDOSKI, J. J.; ELDAWY, A.; MOKBEL, M. F. Streamrec: A real-time recommender system. In: *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: Association for Computing Machinery, 2011. (SIGMOD '11), p. 1243–1246. ISBN 9781450306614. Disponível em: <https://doi.org/10.1145/1989323.1989465>.

CHANG, S.; ZHANG, Y.; TANG, J.; YIN, D.; CHANG, Y.; HASEGAWA-JOHNSON, M. A.; HUANG, T. S. Streaming recommender systems. In: BARRETT, R.; CUMMINGS, R.; AGICHTEIN, E.; GABRILOVICH, E. (Ed.). *Proceedings of the 26th International Conference on World Wide Web, WWW 2017,*

Perth, Australia, April 3-7, 2017. ACM, 2017. p. 381–389. Disponível em: <https://doi.org/10.1145/3038912.3052627>.

CHAUDHURY, S.; YAMASAKI, T. Robustness of adaptive neural network optimization under training noise. *IEEE Access*, v. 9, p. 37039–37053, 2021. Disponível em: <https://doi.org/10.1109/ACCESS.2021.3062990>.

CHEN, J.; FANG, J.; LIU, W.; TANG, T.; YANG, C. clmf: A fine-grained and portable alternating least squares algorithm for parallel matrix factorization. *Future Gener. Comput. Syst.*, v. 108, p. 1192–1205, 2020. Disponível em: <https://doi.org/10.1016/j.future.2018.04.071>.

CHRISTY, A. J.; UMAMAKESWARI, A.; PRIYATHARSINI, L.; NEYAA, A. Rfm ranking–an effective approach to customer segmentation. *Journal of King Saud University-Computer and Information Sciences*, Elsevier, 2018. Disponível em: <https://doi.org/10.1016/j.eswa.2008.07.018>.

CREMONESI, P.; KOREN, Y.; TURRIN, R. Performance of recommender algorithms on top-n recommendation tasks. In: AMATRIAIN, X.; TORRENS, M.; RESNICK, P.; ZANKER, M. (Ed.). *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*. ACM, 2010. p. 39–46. Disponível em: <https://doi.org/10.1145/1864708.1864721>.

CREMONESI, P.; KOREN, Y.; TURRIN, R. Performance of recommender algorithms on top-n recommendation tasks. In: AMATRIAIN, X.; TORRENS, M.; RESNICK, P.; ZANKER, M. (Ed.). *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*. ACM, 2010. p. 39–46. Disponível em: <https://doi.org/10.1145/1864708.1864721>.

DEMSAR, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, v. 7, p. 1–30, 2006. Disponível em: <http://jmlr.org/papers/v7/demsar06a.html>.

DING, J.; FENG, F.; HE, X.; YU, G.; LI, Y.; JIN, D. An improved sampler for bayesian personalized ranking by leveraging view data. In: CHAMPIN, P.; GANDON, F.; LALMAS, M.; IPEIROTIS, P. G. (Ed.). *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*. ACM, 2018. p. 13–14. Disponível em: <https://doi.org/10.1145/3184558.3186905>.

DING, J.; QUAN, Y.; YAO, Q.; LI, Y.; JIN, D. Simplify and robustify negative sampling for implicit collaborative filtering. In: LAROCHELLE, H.; RANZATO, M.; HADSELL, R.; BALCAN, M.; LIN, H. (Ed.). *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. [S.l.: s.n.], 2020.

DOGO, E.; AFOLABI, O.; NWULU, N.; TWALA, B.; AIGBAVBOA, C. A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks. In: IEEE. *2018 international conference on computational techniques, electronics and mechanical systems (CTEMS)*. [S.l.], 2018. p. 92–99.

DOOMS, S.; PESSEMIER, T. D.; MARTENS, L. Mining cross-domain rating datasets from structured data on twitter. In: *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014, Companion Volume*. [S.l.]: ACM, 2014. p. 621–624.

DOZAT, T. Incorporating nesterov momentum into adam. ICLR Workshop, p. 2013–2016, 2016.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937.

GAMA, J. *Knowledge discovery from data streams*. [S.l.]: CRC Press, 2010.

GAMA, J.; SEBASTIÃO, R.; RODRIGUES, P. P. On evaluating stream learning algorithms. *Mach. Learn.*, v. 90, n. 3, p. 317–346, 2013. Disponível em: <https://doi.org/10.1007/s10994-012-5320-9>.

GAMA, J.; ZLIOBAITE, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A. A survey on concept drift adaptation. *ACM Comput. Surv.*, v. 46, n. 4, p. 44:1–44:37, 2014. Disponível em: <https://doi.org/10.1145/2523813>.

GÓMEZ, B. G.; ARRANZ, A. M. G.; CILLÁN, J. G. Drivers of customer likelihood to join grocery retail loyalty programs. an analysis of reward programs and loyalty cards. *Journal of Retailing and Consumer Services*, Elsevier, v. 19, n. 5, p. 492–500, 2012.

GREENBERG-TOLEDO, T.; MAZOR, R.; ALI, A. H.; KVATINSKY, S. Supporting the momentum training algorithm using a memristor-based synapse. *IEEE Trans. Circuits Syst. I Regul. Pap.*, v. 66-I, n. 4, p. 1571–1583, 2019. Disponível em: <https://doi.org/10.1109/TCSI.2018.2888538>.

GUO, G.; QIU, H.; TAN, Z.; LIU, Y.; MA, J.; WANG, X. Resolving data sparsity by multi-type auxiliary implicit feedback for recommender systems. *Knowl. Based Syst.*, v. 138, p. 202–207, 2017. Disponível em: <https://doi.org/10.1016/j.knosys.2017.10.005>.

GUO, L.; YIN, H.; WANG, Q.; CHEN, T.; ZHOU, A.; HUNG, N. Q. V. Streaming session-based recommendation. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*. ACM, 2019. p. 1569–1577. Disponível em: <https://doi.org/10.1145/3292500.3330839>.

HARPER, F. M.; KONSTAN, J. A. The movielens datasets: History and context. *TiiS*, v. 5, n. 4, p. 19:1–19:19, 2016. Disponível em: <https://doi.org/10.1145/2827872>.

HE, X.; CHEN, T.; KAN, M.; CHEN, X. Trirank: Review-aware explainable recommendation by modeling aspects. In: *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*. ACM, 2015. p. 1661–1670. Disponível em: <https://doi.org/10.1145/2806416.2806504>.

HE, X.; LIAO, L.; ZHANG, H.; NIE, L.; HU, X.; CHUA, T. Neural collaborative filtering. In: *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*. [S.l.]: ACM, 2017. p. 173–182.

HE, X.; ZHANG, H.; KAN, M.; CHUA, T. Fast matrix factorization for online recommendation with implicit feedback. In: PEREGO, R.; SEBASTIANI, F.; ASLAM, J. A.; RUTHVEN, I.; ZOBEL, J. (Ed.). *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016*. ACM, 2016. p. 549–558. Disponível em: <https://doi.org/10.1145/2911451.2911489>.

HERLOCKER, J. L.; KONSTAN, J. A.; BORCHERS, A.; RIEDL, J. An algorithmic framework for performing collaborative filtering. In: *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA*. ACM, 1999. p. 230–237. Disponível em: <https://doi.org/10.1145/312624.312682>.

HWANGBO, H.; KIM, Y. S.; CHA, K. J. Recommendation system development for fashion retail e-commerce. *Electron. Commer. Res. Appl.*, v. 28, p. 94–101, 2018. Disponível em: <https://doi.org/10.1016/j.elerap.2018.01.012>.

JORGE, A. M.; VINAGRE, J.; DOMINGUES, M. A.; GAMA, J.; SOARES, C.; MATUSZYK, P.; SPILIOPOULOU, M. Scalable online top-n recommender systems. In: BRIDGE, D.; STUCKENSCHMIDT, H. (Ed.). *E-Commerce and Web Technologies - 17th International Conference, EC-Web 2016, Porto, Portugal, September 5-8, 2016, Revised Selected Papers*. [s.n.], 2016. (Lecture Notes in Business Information Processing, v. 278), p. 3–20. Disponível em: <https://doi.org/10.1007/978-3-319-53676-7_1>.

JOSÉ, E. F.; ENEMBRECK, F.; BARDDAL, J. P. Adadrift: An adaptive learning technique for long-history stream-based recommender systems. In: *Proceedings of IEEE Systems, Man, and Cybernetics 2020 (IEEE SMC 2020)*. [S.l.]: IEEE, 2020.

KASTRATI, M.; BIBA, M. A state-of-the-art survey of advanced optimization methods in machine learning. In: XHINA, E.; HOXHA, K. (Ed.). *Proceedings of the 4th International Conference on Recent Trends and Applications in Computer Science and Information Technology, Tirana, Albania, May 21st - to - 22nd, 2021*. [S.l.]: CEUR-WS.org, 2021. (CEUR Workshop Proceedings, v. 2872), p. 1–10.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. In: BENGIO, Y.; LECUN, Y. (Ed.). *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. [s.n.], 2015. Disponível em: <http://arxiv.org/abs/1412.6980>.

KOREN, Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*. ACM, 2008. p. 426–434. Disponível em: <https://doi.org/10.1145/1401890.1401944>.

KOREN, Y. Collaborative filtering with temporal dynamics. *Commun. ACM*, v. 53, n. 4, p. 89–97, 2010. Disponível em: <http://doi.acm.org/10.1145/1721654.1721677>.

KOREN, Y.; BELL, R. M.; VOLINSKY, C. Matrix factorization techniques for recommender systems. *IEEE Computer*, v. 42, n. 8, p. 30–37, 2009. Disponível em: <https://doi.org/10.1109/MC.2009.263>.

KRISTOFFERSEN, M. S.; SHEPSTONE, S. E.; TAN, Z. A dataset for inferring contextual preferences of users watching TV. In: MITROVIC, T.; ZHANG, J.; CHEN, L.; CHIN, D. (Ed.). *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization, UMAP 2018, Singapore, July 08-11, 2018*. ACM, 2018. p. 367–368. Disponível em: <https://doi.org/10.1145/3209219.3209263>.

LAGHMARI, K.; MARSALA, C.; RAMDANI, M. An adapted incremental graded multi-label classification model for recommendation systems. *Prog. Artif. Intell.*, v. 7, n. 1, p. 15–29, 2018. Disponível em: <https://doi.org/10.1007/s13748-017-0133-5>.

LE, D.; LAUW, H. W.; FANG, Y. Basket-sensitive personalized item recommendation. In: SIERRA, C. (Ed.). *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*. ijcai.org, 2017. p. 2060–2066. Disponível em: <https://doi.org/10.24963/ijcai.2017/286>.

LI, B.; HAN, L. Distance weighted cosine similarity measure for text classification. In: *Intelligent Data Engineering and Automated Learning - IDEAL 2013 - 14th International Conference, IDEAL 2013, Hefei, China, October 20-23, 2013. Proceedings*. [S.l.]: Springer, 2013. (Lecture Notes in Computer Science, v. 8206), p. 611–618.

LI, G.; ZHANG, Z.; WANG, L.; CHEN, Q.; PAN, J. One-class collaborative filtering based on rating prediction and ranking prediction. *Knowl. Based Syst.*, v. 124, p. 46–54, 2017.

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, CIKM 2017, Singapore, November 06 - 10, 2017*. ACM, 2017. p. 1419–1428. Disponível em: <https://doi.org/10.1145/3132847.3132926>.

LIN, C.; WANG, L.; TSAI, K. Hybrid real-time matrix factorization for implicit feedback recommendation systems. *IEEE Access*, v. 6, p. 21369–21380, 2018. Disponível em: <https://doi.org/10.1109/ACCESS.2018.2819428>.

LIU, N. N.; CAO, B.; ZHAO, M.; YANG, Q. Adapting neighborhood and matrix factorization models for context aware recommendation. In: *Proceedings of the Workshop on Context-Aware Movie Recommendation*. [S.l.: s.n.], 2010. p. 7–13.

LUO, X.; QIN, W.; DONG, A.; SEDRAOUI, K.; ZHOU, M. Efficient and high-quality recommendations via momentum-incorporated parallel stochastic gradient descent-based learning. *IEEE CAA J. Autom. Sinica*, v. 8, n. 2, p. 402–411, 2021. Disponível em: <https://doi.org/10.1109/JAS.2020.1003396>.

LUO, X.; XIA, Y.; ZHU, Q. Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowl. Based Syst.*, v. 27, p. 271–280, 2012. Disponível em: <https://doi.org/10.1016/j.knosys.2011.09.006>.

MATUSZYK, P.; VINAGRE, J.; SPILIOPOULOU, M.; JORGE, A. M.; GAMA, J. Forgetting methods for incremental matrix factorization in recommender systems. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*. ACM, 2015. p. 947–953. Disponível em: <https://doi.org/10.1145/2695664.2695820>.

MATUSZYK, P.; VINAGRE, J.; SPILIOPOULOU, M.; JORGE, A. M.; GAMA, J. Forgetting methods for incremental matrix factorization in recommender systems. In: *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*. ACM, 2015. p. 947–953. Disponível em: <https://doi.org/10.1145/2695664.2695820>.

MATUSZYK, P.; VINAGRE, J.; SPILIOPOULOU, M.; JORGE, A. M.; GAMA, J. Forgetting techniques for stream-based matrix factorization in recommender systems. *Knowl. Inf. Syst.*, v. 55, n. 2, p. 275–304, 2018. Disponível em: <https://doi.org/10.1007/s10115-017-1091-8>.

MCAULEY, J. *Amazon Product Data*. 2014. Disponível em: <http://jmcauley.ucsd.edu/data/amazon/links.html>.

NASSAR, N.; JAFAR, A.; RAHHAL, Y. A novel deep multi-criteria collaborative filtering model for recommendation system. *Knowl. Based Syst.*, v. 187, 2020. Disponível em: <https://doi.org/10.1016/j.knosys.2019.06.019>.

NEMENYI, P. B. *Distribution-free multiple comparisons*. Tese (PhD Thesis) — Princeton University, 1963.

NGUYEN, D. M.; TSILIGIANNI, E.; DELIGIANNIS, N. Learning discrete matrix factorization models. *IEEE Signal Process. Lett.*, v. 25, n. 5, p. 720–724, 2018. Disponível em: <https://doi.org/10.1109/LSP.2018.2823268>.

OCEPEK, U.; RUGELJ, J.; BOSNIC, Z. Improving matrix factorization recommendations for examples in cold start. *Expert Syst. Appl.*, v. 42, n. 19, p. 6784–6794, 2015. Disponível em: <https://doi.org/10.1016/j.eswa.2015.04.071>.

PAN, R.; ZHOU, Y.; CAO, B.; LIU, N. N.; LUKOSE, R. M.; SCHOLZ, M.; YANG, Q. One-class collaborative filtering. In: *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*. [S.l.]: IEEE Computer Society, 2008. p. 502–511.

PAN, W.; LIU, M.; MING, Z. Transfer learning for heterogeneous one-class collaborative filtering. *IEEE Intell. Syst.*, v. 31, n. 4, p. 43–49, 2016.

PANDEY, A. K.; RAJPOOT, D. S. Resolving cold start problem in recommendation system using demographic approach. In: IEEE. *2016 International Conference on Signal Processing and Communication (ICSC)*. 2016. p. 213–218. Disponível em: <https://doi.org/10.1109/ICSPCom.2016.7980578>.

PIRAMUTHU, S.; KAPOOR, G.; ZHOU, W.; MAUW, S. Input online review data and related bias in recommender systems. *Decis. Support Syst.*, v. 53, n. 3, p. 418–424, 2012. Disponível em: <https://doi.org/10.1016/j.dss.2012.02.006>.

PORTUGAL, I.; ALENCAR, P. S. C.; COWAN, D. D. The use of machine learning algorithms in recommender systems: A systematic review. *Expert Syst. Appl.*, v. 97, p. 205–227, 2018. Disponível em: <https://doi.org/10.1016/j.eswa.2017.12.020>.

RABIU, I.; SALIM, N.; DA'U, A.; OSMAN, A.; NASSER, M. Exploiting dynamic changes from latent features to improve recommendation using temporal matrix factorization. *Egyptian Informatics Journal*, Elsevier, 2020.

RAGHUWANSHI, S. K.; PATERIYA, R. K. Accelerated singular value decomposition (asvd) using momentum based gradient descent optimization. *Journal of King Saud University-Computer and Information Sciences*, Elsevier, 2018.

RANA, C.; JAIN, S. K. An evolutionary clustering algorithm based on temporal features for dynamic recommender systems. *Swarm Evol. Comput.*, v. 14, p. 21–30, 2014. Disponível em: <https://doi.org/10.1016/j.swevo.2013.08.003>.

REDDI, S. J.; KALE, S.; KUMAR, S. On the convergence of adam and beyond. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. [S.l.]: OpenReview.net, 2018.

RENDLE, S.; FREUDENTHALER, C. Improving pairwise learning for item recommendation from implicit feedback. In: CARTERETTE, B.; DIAZ, F.; CASTILLO, C.; METZLER, D. (Ed.). *Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, February 24-28, 2014*. ACM, 2014. p. 273–282. Disponível em: <https://doi.org/10.1145/2556195.2556248>.

RENDLE, S.; FREUDENTHALER, C.; GANTNER, Z.; SCHMIDT-THIEME, L. BPR: bayesian personalized ranking from implicit feedback. *CoRR*, abs/1205.2618, 2012.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Introduction to recommender systems handbook. In: *Recommender Systems Handbook*. Springer, 2011. p. 1–35. Disponível em: <https://doi.org/10.1007/978-0-387-85820-3_1>.

SALAKHUTDINOV, R.; MNIH, A. Probabilistic matrix factorization. In: PLATT, J. C.; KOLLER, D.; SINGER, Y.; ROWEIS, S. T. (Ed.). *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. [S.l.]: Curran Associates, Inc., 2007. p. 1257–1264.

SARWAR, B. M.; KARYPIS, G.; KONSTAN, J. A.; RIEDL, J. Item-based collaborative filtering recommendation algorithms. In: *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*. ACM, 2001. p. 285–295. Disponível em: <https://doi.org/10.1145/371920.372071>.

SHAO, B.; LI, X.; BIAN, G. A survey of research hotspots and frontier trends of recommendation systems from the perspective of knowledge graph. *Expert Systems with Applications*, Elsevier, v. 165, p. 113764, 2021. Disponível em: <https://doi.org/10.1016/j.eswa.2020.113764>.

SHI, Y.; LARSON, M. A.; HANJALIC, A. List-wise learning to rank with matrix factorization for collaborative filtering. In: *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*. [S.l.]: ACM, 2010. p. 269–272.

SIDANA, S.; LACLAU, C.; AMINI, M.; VANDELLE, G.; BOIS-CRETTEZ, A. KASANDR: A large-scale dataset with implicit feedback for recommendation. In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. ACM, 2017. p. 1245–1248. Disponível em: <https://doi.org/10.1145/3077136.3080713>.

SILVA, N.; CARVALHO, D.; PEREIRA, A. C. M.; MOURÃO, F.; ROCHA, L. C. da. The pure cold-start problem: A deep study about how to conquer first-time users in recommendations domains. *Inf. Syst.*, v. 80, p. 1–12, 2019. Disponível em: <https://doi.org/10.1016/j.is.2018.09.001>.

SONG, B.; YANG, X.; CAO, Y.; XU, C. Neural collaborative ranking. In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*. [S.l.]: ACM, 2018. p. 1353–1362.

SONG, Q.; CHENG, J.; LU, H. Incremental matrix factorization via feature space re-learning for recommender system. In: WERTHNER, H.; ZANKER, M.; GOLBECK, J.; SEMERARO, G. (Ed.). *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015*. ACM, 2015. p. 277–280. Disponível em: <https://doi.org/10.1145/2792838.2799668>.

SUN, S.; CAO, Z.; ZHU, H.; ZHAO, J. A survey of optimization methods from a machine learning perspective. *IEEE Trans. Cybern.*, v. 50, n. 8, p. 3668–3681, 2020.

TAHMASEBI, F.; MEGHDADI, M.; AHMADIAN, S.; VALIALLAHI, K. A hybrid recommendation system based on profile expansion technique to alleviate cold start problem. *Multimedia Tools and Applications*, Springer, p. 1–16, 2020. Disponível em: <https://doi.org/10.1007/s11042-020-09768-8>.

TAKÁCS, G.; PILÁSZY, I.; NÉMETH, B.; TIKK, D. Scalable collaborative filtering approaches for large recommender systems. *J. Mach. Learn. Res.*, v. 10, p. 623–656, 2009. Disponível em: <https://dl.acm.org/citation.cfm?id=1577091>.

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, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*. ACM, 2016. p. 17–22. Disponível em: <https://doi.org/10.1145/2988450.2988452>.

TATIANA, K.; MIKHAIL, M. Market basket analysis of heterogeneous data sources for recommendation system improvement. *Procedia Computer Science*, Elsevier, v. 136, p. 246–254, 2018.

TIELEMAN, T.; HINTON, G. *Lecture 6.5 - RMSProp, COURSERA: Neural Networks for Machine Learning*. [S.l.], 2012.

TSYMBAL, A. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, Citeseer, v. 106, n. 2, p. 58, 2004.

ULLAH, F.; SARWAR, G.; LEE, S. C.; PARK, Y. K.; MOON, K.; KIM, J. T. Hybrid recommender system with temporal information. In: *2012 International Conference on Information Networking, ICOIN 2012, Bali, Indonesia, February 1-3, 2012*. IEEE Computer Society, 2012. p. 421–425. Disponível em: <https://doi.org/10.1109/ICOIN.2012.6164413>.

VINAGRE, J.; JORGE, A. M.; GAMA, J. Fast incremental matrix factorization for recommendation with positive-only feedback. In: *User Modeling, Adaptation, and Personalization - 22nd International Conference, UMAP 2014, Aalborg,*

Denmark, July 7-11, 2014. Proceedings. Springer, 2014. (Lecture Notes in Computer Science, v. 8538), p. 459–470. Disponível em: <https://doi.org/10.1007/978-3-319-08786-3_41>.

VINISKI, A. D.; BARDDAL, J. P.; BRITTO JR., A. de S. UKIRF: an item rejection framework for improving negative items sampling in one-class collaborative filtering. In: *Advances in Knowledge Discovery and Data Mining - 25th Pacific-Asia Conference, PAKDD 2021, Virtual Event, May 11-14, 2021, Proceedings, Part II*. Springer, 2021. (Lecture Notes in Computer Science, v. 12713), p. 549–560. Disponível em: <https://doi.org/10.1007/978-3-030-75765-6_44>.

VINISKI, A. D.; BARDDAL, J. P.; BRITTO JR., A. de S.; ENEMBRECK, F.; CAMPOS, H. V. A. de. A case study of batch and incremental recommender systems in supermarket data under concept drifts and cold start. *Expert Systems with Applications*, v. 176, p. 114890, 2021. ISSN 0957-4174. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0957417421003316>.

VOLKOVS, M.; YU, G. W. Effective latent models for binary feedback in recommender systems. In: *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*. [S.l.]: ACM, 2015. p. 313–322.

WANG, P.; GUO, J.; LAN, Y. Modeling retail transaction data for personalized shopping recommendation. In: LI, J.; WANG, X. S.; GAROFALAKIS, M. N.; SOBOROFF, I.; SUEL, T.; WANG, M. (Ed.). *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*. ACM, 2014. p. 1979–1982. Disponível em: <https://doi.org/10.1145/2661829.2662020>.

WANG, P.; GUO, J.; LAN, Y.; XU, J.; WAN, S.; CHENG, X. Learning hierarchical representation model for nextbasket recommendation. In: *Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval*. [S.l.: s.n.], 2015. p. 403–412.

WEBB, G. I.; LEE, L. K.; GOETHALS, B.; PETITJEAN, F. Analyzing concept drift and shift from sample data. *Data Mining and Knowledge Discovery*, Springer, v. 32, n. 5, p. 1179–1199, 2018.

WEI, J.; HE, J.; CHEN, K.; ZHOU, Y.; TANG, Z. Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Syst. Appl.*, v. 69, p. 29–39, 2017. Disponível em: <https://doi.org/10.1016/j.eswa.2016.09.040>.

WENG, C. Revenue prediction by mining frequent itemsets with customer analysis. *Eng. Appl. Artif. Intell.*, v. 63, p. 85–97, 2017. Disponível em: <https://doi.org/10.1016/j.engappai.2017.04.020>.

WU, H.; WANG, Y.; CHENG, X. Incremental probabilistic latent semantic analysis for automatic question recommendation. In: *Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys 2008, Lausanne, Switzerland, October 23-25, 2008*. ACM, 2008. p. 99–106. Disponível em: <https://doi.org/10.1145/1454008.1454026>.

XU, Y.; ZHANG, Y.; GUO, W.; GUO, H.; TANG, R.; COATES, M. Graphsail: Graph structure aware incremental learning for recommender systems. In: *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*. ACM, 2020. p. 2861–2868. Disponível em: <https://doi.org/10.1145/3340531.3412754>.

YANG, M.; DAI, Q.; DONG, Z.; CHEN, X.; HE, X.; WANG, J. Top-n recommendation with counterfactual user preference simulation. In: DEMARTINI, G.; ZUCCON, G.; CULPEPPER, J. S.; HUANG, Z.; TONG, H. (Ed.). *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*. ACM, 2021. p. 2342–2351. Disponível em: <https://doi.org/10.1145/3459637.3482305>.

YAO, Y.; TONG, H.; YAN, G.; XU, F.; ZHANG, X.; SZYMANSKI, B. K.; LU, J. Dual-regularized one-class collaborative filtering with implicit feedback. *World Wide Web*, v. 22, n. 3, p. 1099–1129, 2019.

YE, J. Cosine similarity measures for intuitionistic fuzzy sets and their applications. *Math. Comput. Model.*, v. 53, n. 1-2, p. 91–97, 2011.

YIN, R.; LI, K.; ZHANG, G.; LU, J. A deeper graph neural network for recommender systems. *Knowl. Based Syst.*, v. 185, 2019. Disponível em: <https://doi.org/10.1016/j.knosys.2019.105020>.

YOU, J.; WANG, Y.; PAL, A.; EKSOMBATCHAI, P.; ROSENBERG, C.; LESKOVEC, J. Hierarchical temporal convolutional networks for dynamic recommender systems. In: *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. ACM, 2019. p. 2236–2246. Disponível em: <https://doi.org/10.1145/3308558.3313747>.

YU, H.; BILENKO, M.; LIN, C. Selection of negative samples for one-class matrix factorization. In: *Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, Texas, USA, April 27-29, 2017*. [S.l.]: SIAM, 2017. p. 363–371.

YU, T.; MENGSHOEL, O. J.; JUDE, A.; FELLER, E.; FORGEAT, J.; RADIA, N. Incremental learning for matrix factorization in recommender systems. In: *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*. IEEE Computer Society, 2016. p. 1056–1063. Disponível em: <https://doi.org/10.1109/BigData.2016.7840707>.

YU, Y.; LIU, F. Effective neural network training with a new weighting mechanism-based optimization algorithm. *IEEE Access*, v. 7, p. 72403–72410, 2019. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2919987>.

YUAN, Q.; CHEN, L.; ZHAO, S. Factorization vs. regularization: fusing heterogeneous social relationships in top-n recommendation. In: *Proceedings of the 2011 ACM Conference on Recommender Systems, RecSys 2011, Chicago, IL, USA, October 23-27, 2011*. ACM, 2011. p. 245–252. Disponível em: <https://doi.org/10.1145/2043932.2043975>.

ZAKARIA, I.; RAHMAN, B. A.; OTHMAN, A. K.; YUNUS, N. A. M.; DZULKIPLI, M. R.; OSMAN, M. A. F. The relationship between loyalty program, customer satisfaction and customer loyalty in retail industry: A case study. *Procedia-Social and Behavioral Sciences*, Elsevier, v. 129, p. 23–30, 2014.

ZHANG, J.; LU, X. A multi-trans matrix factorization model with improved time weight in temporal recommender systems. *IEEE Access*, v. 8, p. 2408–2416, 2020. Disponível em: <https://doi.org/10.1109/ACCESS.2019.2960540>.

ZHANG, S.; YAO, L.; SUN, A.; TAY, Y. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, v. 52, n. 1, p. 5:1–5:38, 2019.

ZHANG, W.; CHEN, T.; WANG, J.; YU, Y. Optimizing top-n collaborative filtering via dynamic negative item sampling. In: JONES, G. J. F.; SHERIDAN, P.; KELLY, D.; RIJKE, M. de; SAKAI, T. (Ed.). *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*. ACM, 2013. p. 785–788. Disponível em: <https://doi.org/10.1145/2484028.2484126>.

ZHANG, Z.; LIU, Y.; JIN, Z.; ZHANG, R. A dynamic trust based two-layer neighbor selection scheme towards online recommender systems. *Neurocomputing*, v. 285, p. 94–103, 2018. Disponível em: <https://doi.org/10.1016/j.neucom.2017.12.063>.

ZHENG, X.; LUO, Y.; SUN, L.; ZHANG, J.; CHEN, F. A tourism destination recommender system using users' sentiment and temporal dynamics. *J. Intell. Inf. Syst.*, v. 51, n. 3, p. 557–578, 2018. Disponível em: <https://doi.org/10.1007/s10844-018-0496-5>.

ZHOU, H.; HIRASAWA, K. Evolving temporal association rules in recommender system. *Neural Comput. Appl.*, v. 31, n. 7, p. 2605–2619, 2019. Disponível em: <https://doi.org/10.1007/s00521-017-3217-z>.

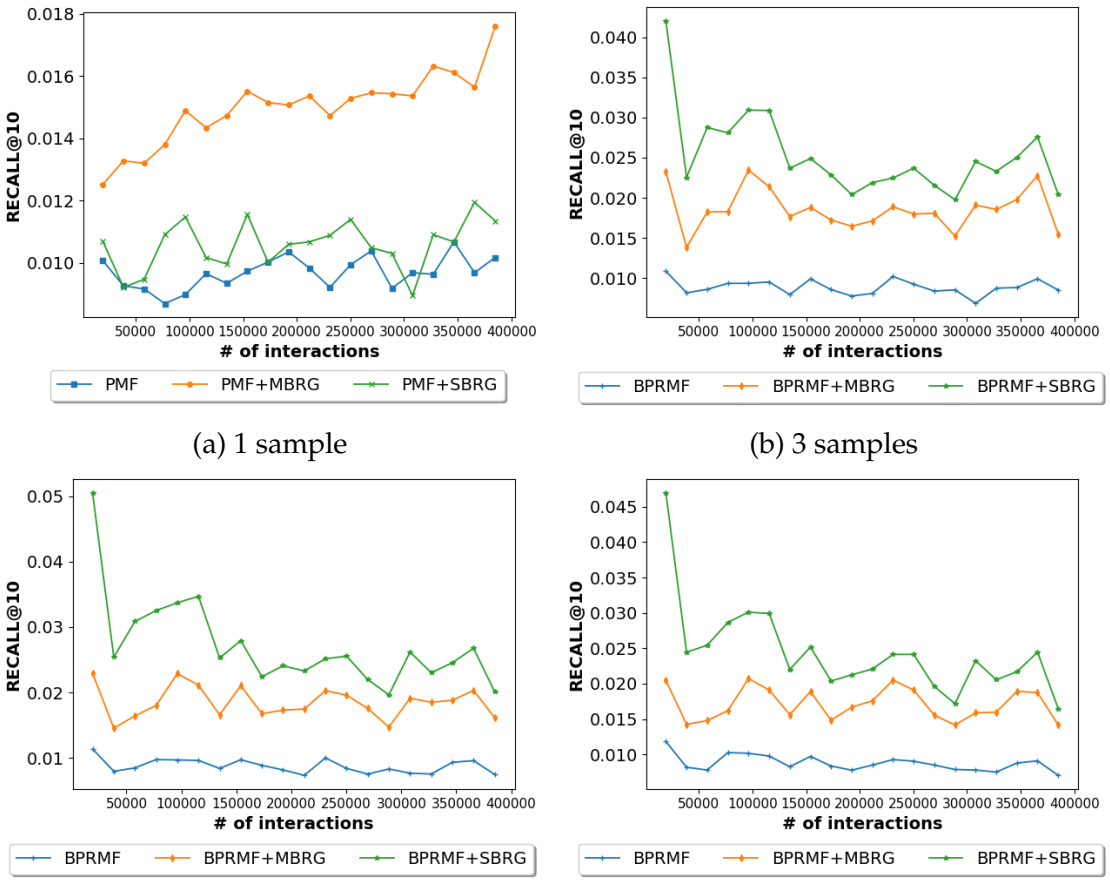
ZHOU, W.; ZHOU, Y.; LI, J.; MEMON, M. H. Lsrec: Large-scale social recommendation with online update. *Expert Syst. Appl.*, v. 162, p. 113739, 2020. Disponível em: <https://doi.org/10.1016/j.eswa.2020.113739>.

ZIKOPOULOS, P.; EATON, C. *Understanding big data: Analytics for enterprise class hadoop and streaming data*. [S.l.]: McGraw-Hill Osborne Media, 2011.

A

Windowed Evaluation of the Negative-Relevant Proposed Strategies

This appendix presents all the generated images related to the negative-relevant sampling strategies. These images are part of the windowed evaluation and show the results of RECALL@10 values considering the number of selected samples per positive instance, datasets, and models. First, Figures A.1, A.2, A.3, A.4, and A.5 present the results obtained by the BPRMF model variant in the Amazon Books, Movie Lens 1M, Movie Tweetings, SMDI-200UE, and TaFeng datasets, respectively. Next, the results of the PMF variants are presented in Figures A.6, A.7, A.8, A.9, and A.10



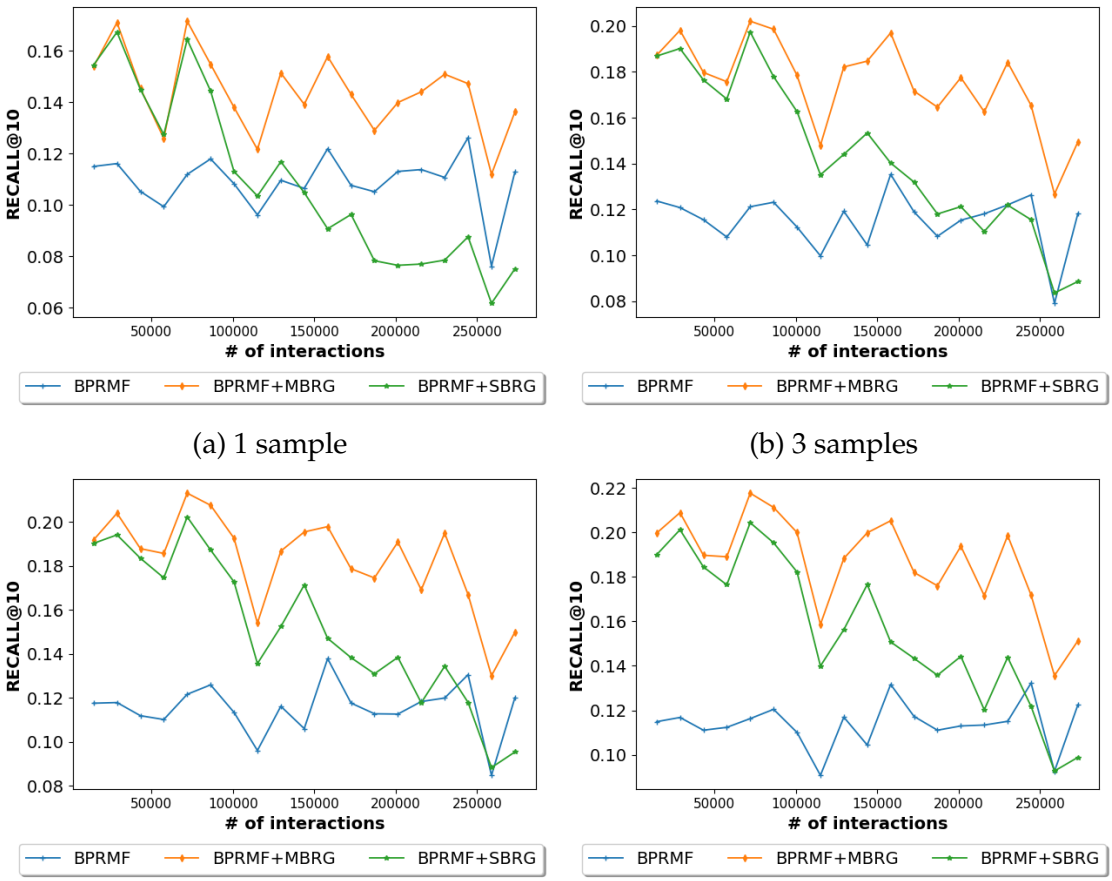
(a) 1 sample

(b) 3 samples

(c) 5 samples

(d) Amazon Books - 10 samples

Figure A.1: Windowed Evaluation of RECALL@10 values for the experiments of the BPRMF model variants in the Amazon Books dataset.



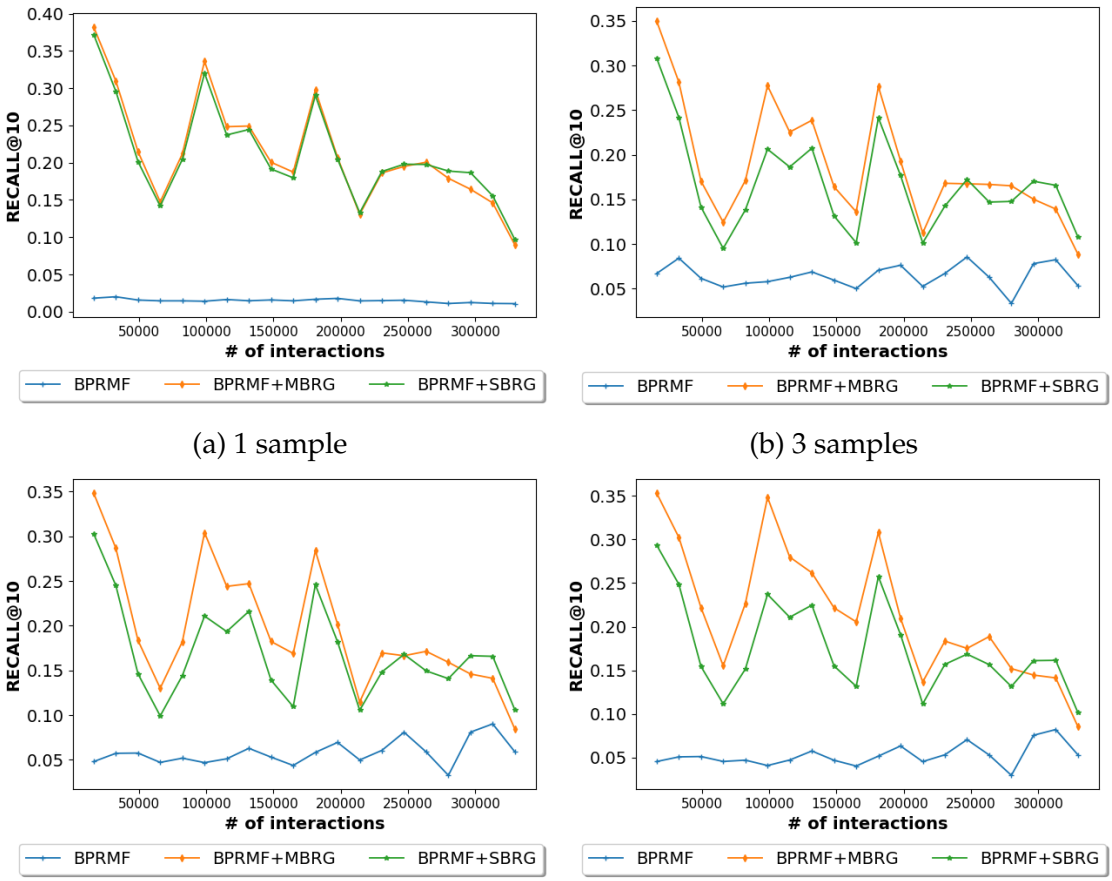
(a) 1 sample

(b) 3 samples

(c) 5 samples

(d) 10 samples

Figure A.2: Windowed Evaluation of RECALL@10 values for the experiments of the BPRMF model variants in the Movie Lens 1M dataset.



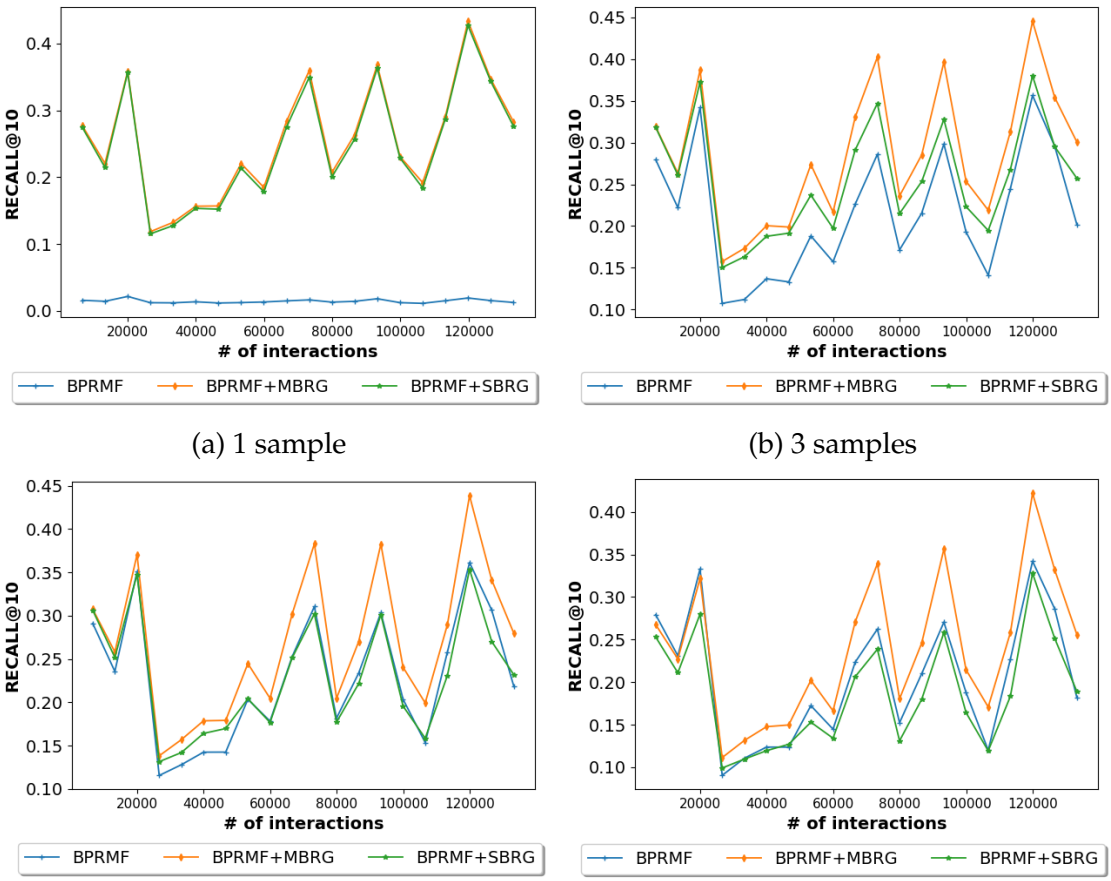
(a) 1 sample

(b) 3 samples

(c) 5 samples

(d) 10 samples

Figure A.3: Windowed Evaluation of RECALL@10 values for the experiments of the BPRMF model variants in the Movie Tweetings dataset.



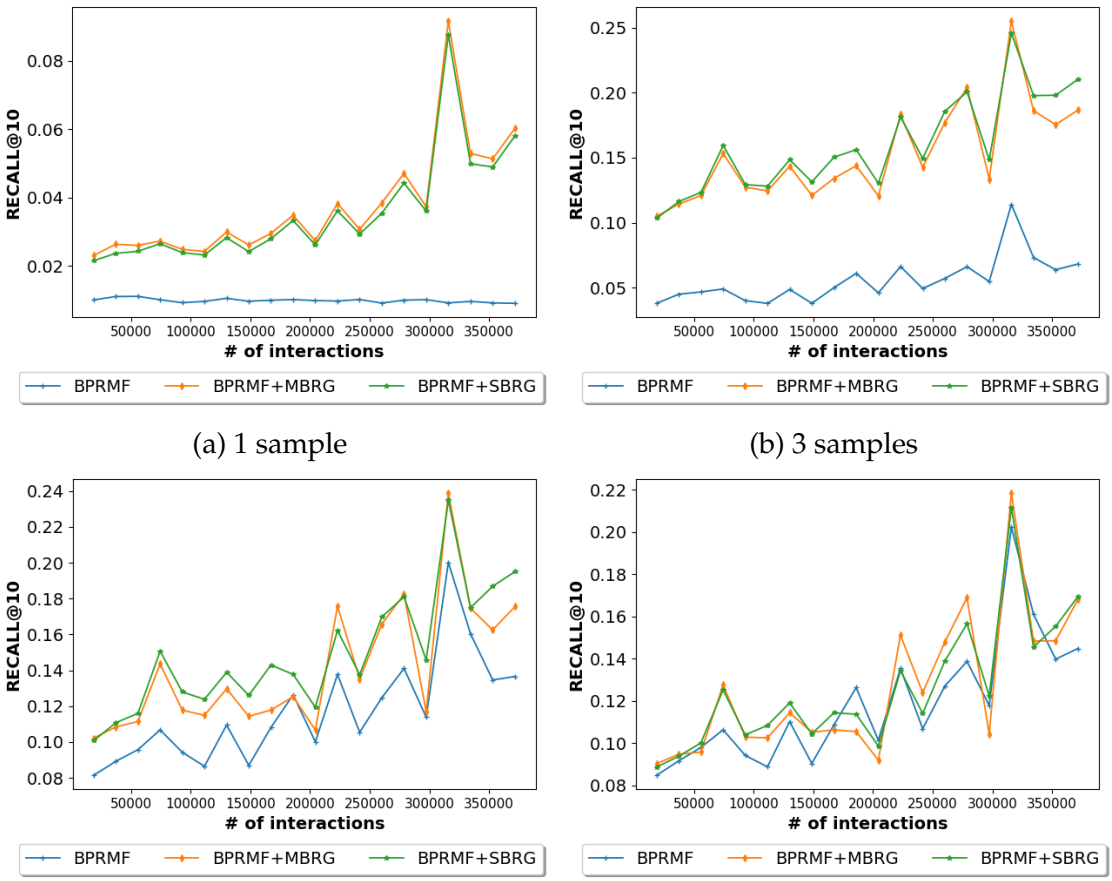
(a) 1 sample

(b) 3 samples

(c) 5 samples

(d) 10 samples

Figure A.4: Windowed Evaluation of RECALL@10 values for the experiments of the BPRMF model variants in the SMDI-200UE dataset.



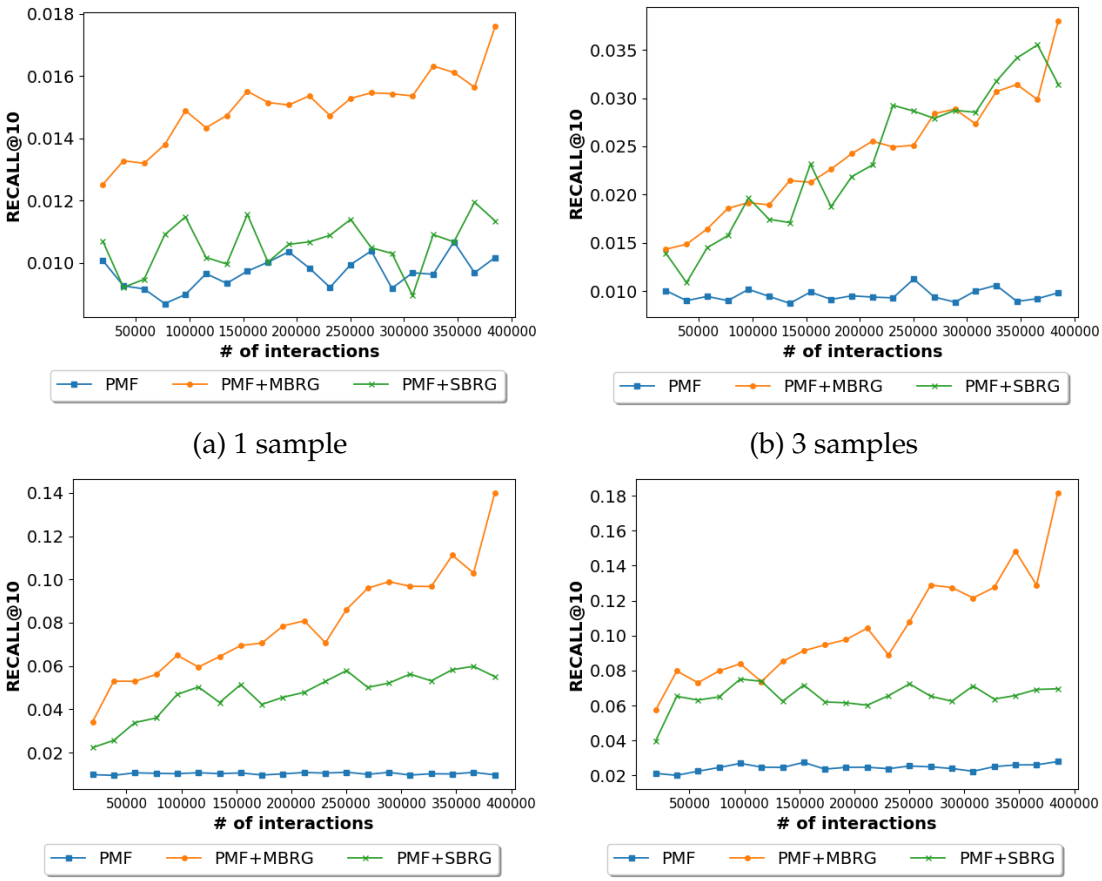
(a) 1 sample

(b) 3 samples

(c) 5 samples

(d) 10 samples

Figure A.5: Windowed Evaluation of RECALL@10 values for the experiments of the BPRMF model variants in the TaFeng dataset.



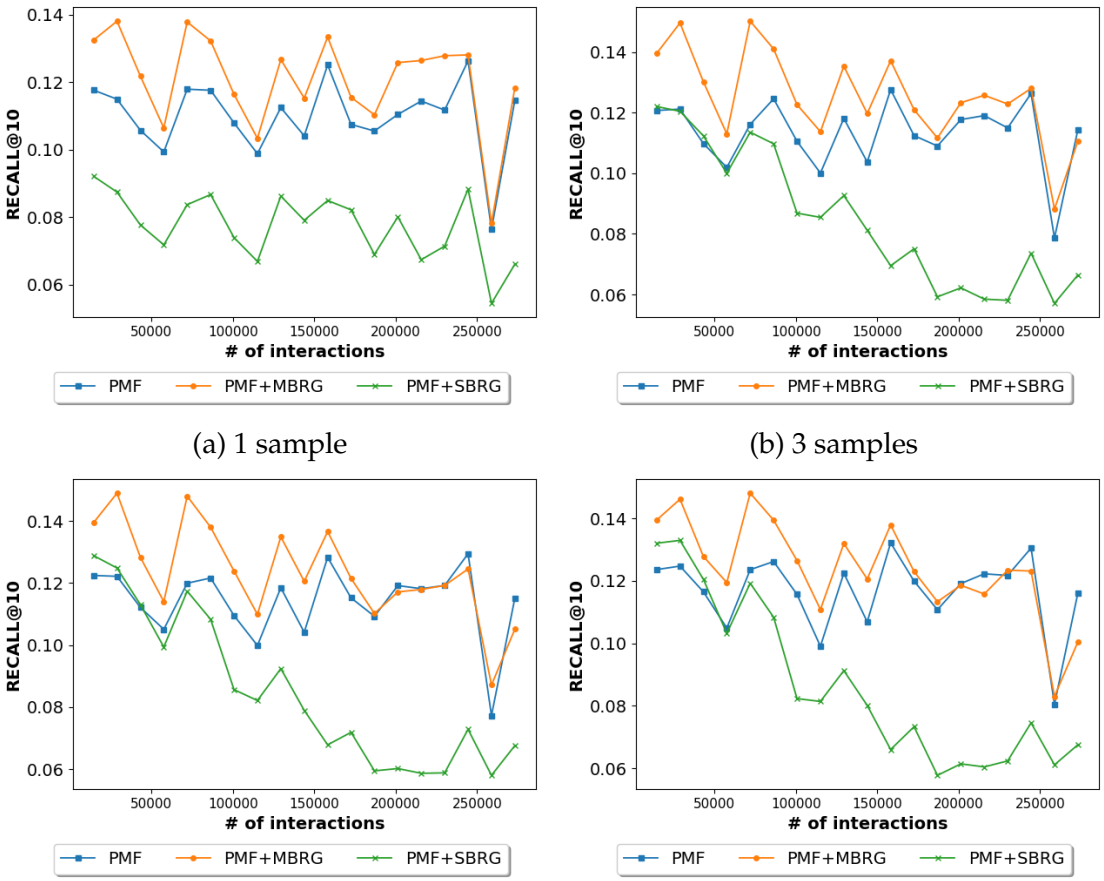
(a) 1 sample

(b) 3 samples

(c) 5 samples

(d) Amazon Books - 10 samples

Figure A.6: Windowed Evaluation of RECALL@10 values for the experiments of the PMF model variants in the Amazon Books dataset.



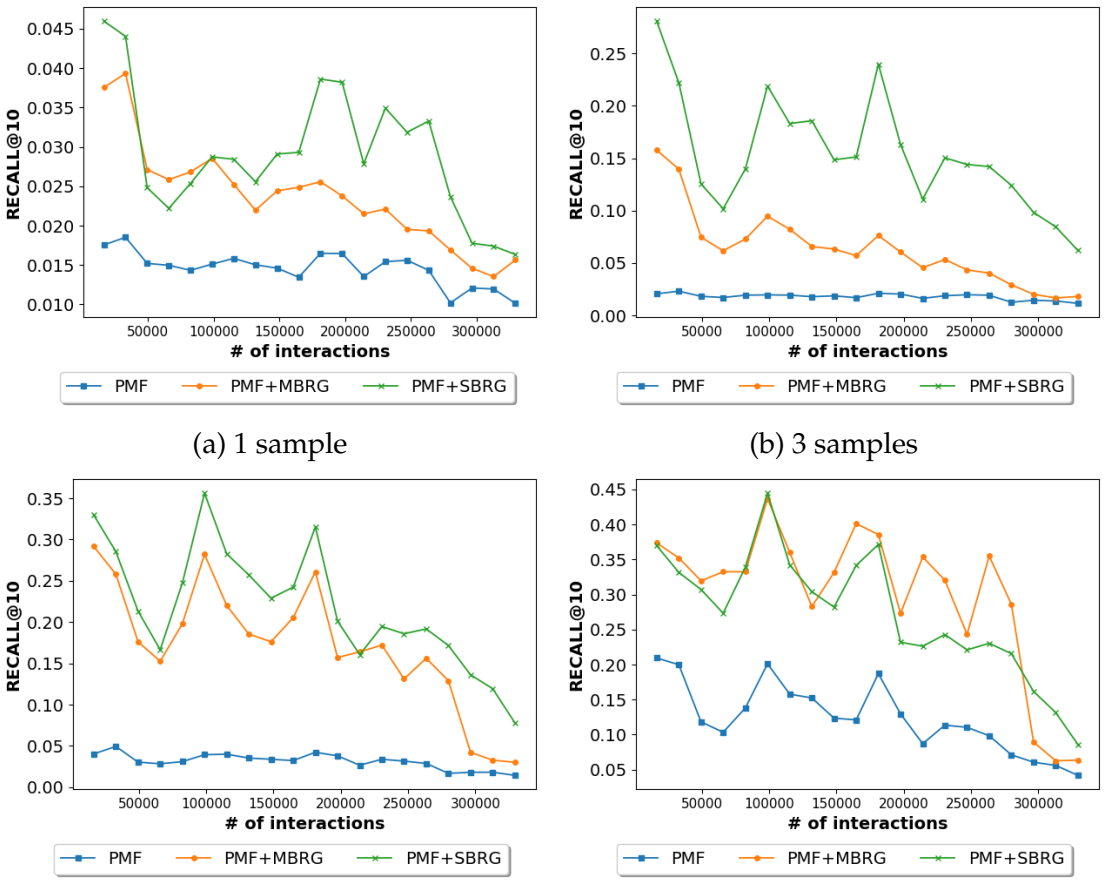
(a) 1 sample

(b) 3 samples

(c) 5 samples

(d) 10 samples

Figure A.7: Windowed Evaluation of RECALL@10 values for the experiments of the PMF model variants in the Movie Lens 1M dataset.



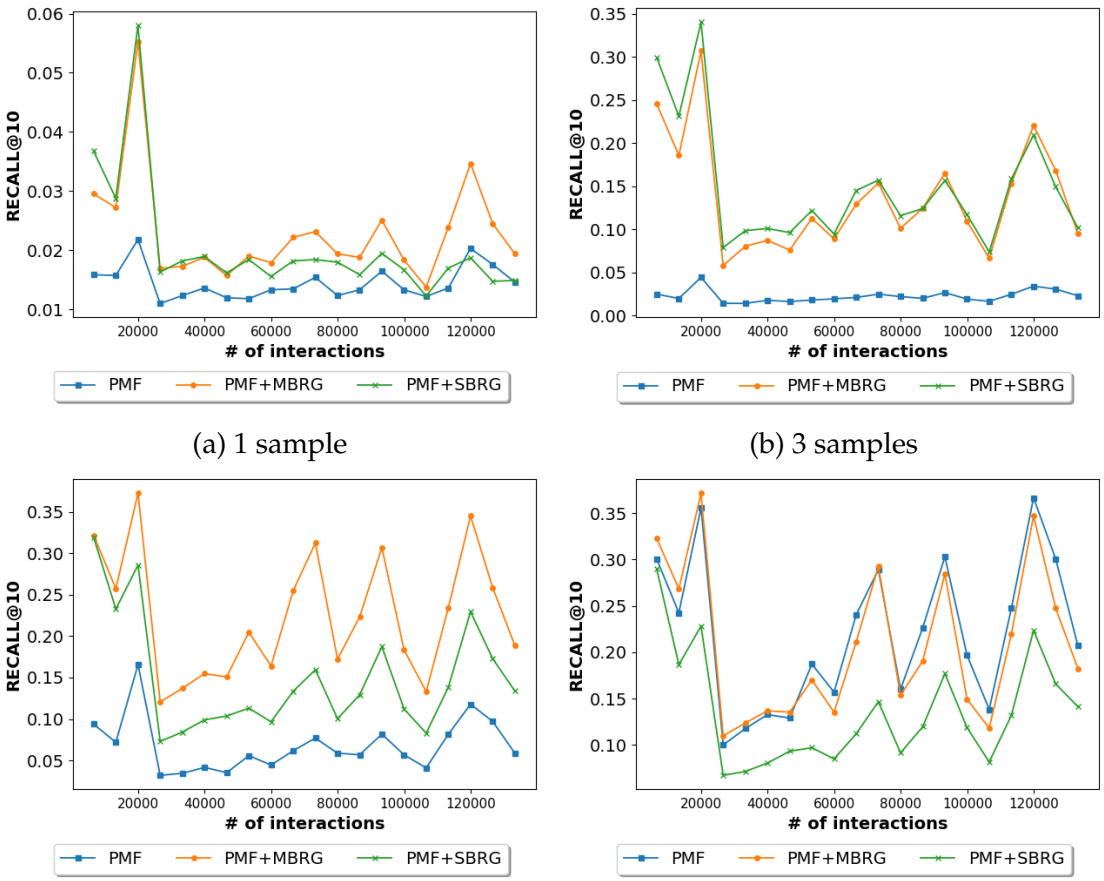
(a) 1 sample

(b) 3 samples

(c) 5 samples

(d) 10 samples

Figure A.8: Windowed Evaluation of RECALL@10 values for the experiments of the PMF model variants in the Movie Tweetings dataset.



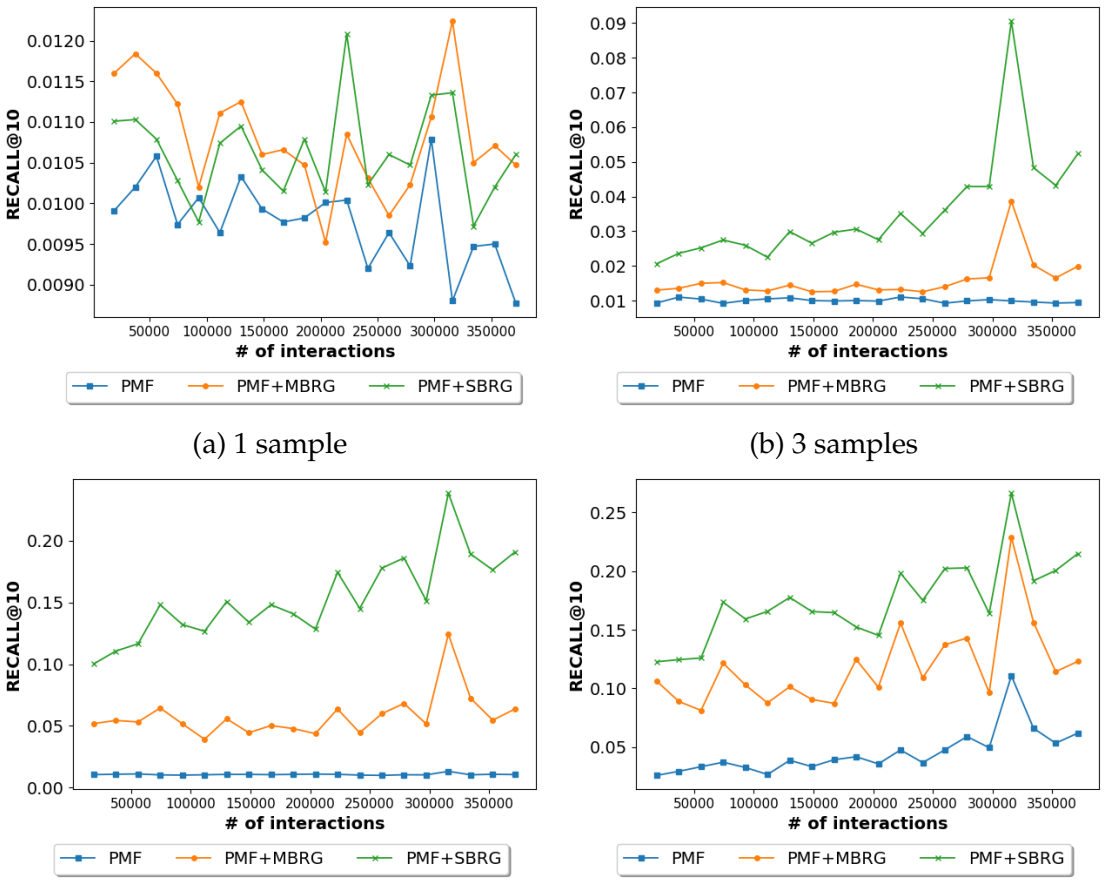
(a) 1 sample

(b) 3 samples

(c) 5 samples

(d) 10 samples

Figure A.9: Windowed Evaluation of RECALL@10 values for the experiments of the PMF model variants in the SMDI-200UE dataset.



(a) 1 sample

(b) 3 samples

(c) 5 samples

(d) 10 samples

Figure A.10: Windowed Evaluation of RECALL@10 values for the experiments of the PMF model variants in the TaFeng dataset.