UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JÚLIO BARRETO GUEDES DA COSTA

# AN INVESTIGATION OF PRE-TRAINING APPROACHES FOR MATRIX FACTORIZATION-BASED RECOMMENDER SYSTEMS

CAMPINA GRANDE
2024

Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Coordenação de Pós-Graduação em Ciência da Computação

# An Investigation of Pre-Training Approaches for Matrix Factorization-based Recommender Systems

Júlio Barreto Guedes da Costa

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação

Linha de Pesquisa: Linha de Pesquisa

Leandro Balby Marinho

(Orientador)

Campina Grande, Paraíba, Brasil

C837c Costa, Júlio Barreto Guedes da.
An investigation of pre-training approaches for matrix factorization-based recommender systems / Júlio Barreto Guedes da Costa. – Campina Grande, 2024.
92 f. : il. color.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2024.
"Orientação: Prof. Dr. Leandro Balby Marinho".
Referências.

1. Aprendizagem de Máquina. 2. Sistemas de Recomendação. 3. Transferência de Aprendizado. I. Marinho, Leandro Balby. II. Título.

CDU 004.85(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA SEVERINA SUELI DA SILVA OLIVEIRA CRB-15/225

MINISTÉRIO DA EDUCAÇÃO
**UNIVERSIDADE FEDERAL DE CAMPINA GRANDE**
POS-GRADUACAO EM CIENCIA DA COMPUTACAO
Rua Aprígio Veloso, 882, Edifício Telmo Silva de Araújo, Bloco CG1, - Bairro Universitário, Campina Grande/PB, CEP 58429-900
Telefone: 2101-1122 - (83) 2101-1123 - (83) 2101-1124
Site: http://computacao.ufcg.edu.br - E-mail: secretaria-copin@computacao.ufcg.edu.br / copin@copin.ufcg.edu.br

**FOLHA DE ASSINATURA PARA TESES E DISSERTAÇÕES**

**JÚLIO BARRETO GUEDES DA COSTA**

**AN INVESTIGATION OF PRE-TRAINING APPROACHES FOR MATRIX FACTORIZATION-BASED RECOMMENDER SYSTEMS**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação como pré-requisito para obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 05/04/2024

Prof. Dr. LEANDRO BALBY MARINHO, UFCG, Orientador

Prof. Dr. CLÁUDIO ELÍZIO CALAZANS CAMPELO, UFCG, Examinador Interno

Prof. Dr. MARCELO GARCIA MANZATO, IFPB, Examinador Externo

# Resumo

Sistemas de Recomendação (RSs, *Recommender Systems*) consistuem um campo de pesquisa e aplicação cujo objetivo é recuperar itens relevantes dado o histórico de interesses anteriores de um usuário. Desde o desafio aberto proposto pela Netflix para melhoria de performance em recomendação de filmes (*Netflix Prize*), RSs utilizam fatores latentes, ou *embeddings*, inicializados aleatoriamente e atualizados durante as etapas de treinamento, como representação para ambos usuários e itens. Observando o grande campo da Aprendizagem de Máquina (*ML, Machine Learning*), diferentes áreas de aplicação obtiveram melhoria de performance através da transferência de aprendizagem (*Transfer Learning*), à exemplo da grande evolução obtida nas tarefas relacionadas à Visão Computacional (*CV, Computer Vision*) após a introdução de modelos como *VGG* e *AlexNet*, mas também muito presente em tarefas do campo de Processamento de Linguagem Natural (*NLP, Natural Language Processing*), especialmente com a popularização dos Grandes Modelos de Linguagem (*LLMs, Large Language Models*) como as famílias de modelos *BERT* e, mais popular recentemente, *GPT*. Diferente das outras áreas de aplicação, entretanto, *Transfer Learning* em RSs não é trivial, visto que as entidades geralmente se restringem à usuários e itens, enquanto em CV, as entidades são imagens e, em NLP, são palavras. O objetivo desta pesquisa é, portanto, estudar possíveis aplicações de *Transfer Learning* em modelos de recomendação, avaliando como diferentes inicializações impactam a performance preditiva de um modelo, através de técnicas de não-supervisionadas, auto-supervisionadas, e supervisionadas.

# Abstract

Recommender Systems (RSs) consist of a field of research and application with the goal of retrieving relevant items for a user. Since the open Netflix Prize challenge for performance improvement in RSs, they have constantly been built by representing users and items as latent factors, more commonly known as *embeddings*, which are often randomly initialized and updated during the training stages. When looking at the greater Machine Learning (ML) area, different areas of application obtained performance improvement through Transfer Learning, such as the boost obtained in the Computer Vision (CV) tasks after the proposal of models like *VGG* or *AlexNet*, or the one achieved in Natural Language Processing (NLP) tasks, especially after the popularization of Large Language Models (LLMs) such as the *BERT* and, more recently, the *GPT* model families. Unlike other application areas, however, Transfer Learning for RSs is not trivial since users and items are the entities, while in CV and NLP, the entities are images and words, respectively. This research aims to study possible applications of Transfer Learning for RSs, evaluating how unsupervised, self-supervised, and supervised embedding initialization impact the predictive performance of the models.

# Agradecimentos

Agradeço inicialmente aos meus pais, Edilson e Tamar, por todo o apoio que me deram até aqui, mas especialmente por incentivar e possibilitar, no máximo dos seus esforços, para que eu tivesse acesso à educação de qualidade. Agradeço também as minhas irmãs, Thaís, Elloá e Laura, que sempre me foram exemplos de pessoas e profissionais.

Gostaria também de agradecer ao meu orientador, Leandro, por me guiar como pesquisador e profissional. O tempo que trabalhamos juntos certamente será fundamental para a construção do resto da minha carreira. Agradeço também aos professores Rodrygo Santos e Denis Parra, cujas contribuições na produção de artigos ajudaram a direcionar e ampliar minha perspectiva sobre a pesquisa. Nos momentos de maior dificultade no decorrer dessa pesquisa, sempre pude contar com o apoio de Elloá e de Iann, sem eles eu não teria conseguido tanto.

Ao Laboratório de Mineração de Dados (LMD) e ao Laboratório de Sistemas Distribuídos (LSD), que me possibilitaram conhecer e trabalhar com diversos profissionais e pesquisadores, por proverem parte da infraestrutura necessária para execução dos experimentos, e por serem, em conjunto, a minha casa dentro da UFCG. Um dos projetos de pesquisa também me possibilitou iniciar a vida profissional, e sou muito grato ao time do Indaband, em especial à Helielson Santos e Andrews Medina, que acreditaram em mim e me convidaram para fazer parte dessa equipe sensacional com um objetivo brilhante.

Além do âmbito profissional, durante esses anos também melhorei bastante como pessoa. Ainda na Paraíba, agradeço à Newton, Gustavo e, novamente, Iann, meus amigos que também são irmãos, por todas as risadas e momentos compartilhados durante todos esses anos de amizade. São Paulo me permitiu aproximar de muitas novas pessoas, então gostaria de agradecer à Thales e Fernando, e, mais recentemente, Ernane, Chen e Leo, que se tornaram minha “casa fora de casa”.

Por fim, mas não menos importante, gostaria de agradecer à Kunumi, pelo suporte financeiro para que esta pesquisa fosse executada durante um período em que a educação pública brasileira estava sob ataque. Zelar por um ensino público de qualidade e manutenção das instituições de pesquisa é nosso dever como cientistas brasileiros.

# Contents

# List of Symbols

CV - *Computer Vision*

CF - *Collaborative Filtering*

DL - *Deep Learning*

DNN - *Deep Neural Network*

MF - *Matrix Factorization*

ML - *MovieLens*

NLP - *Natural Language Processing*

NN - *Neural Network*

OOV - *Out of Vocabulary*

PCA - *Principal Component Analysis*

RS - *Recommender System*

SG - *SkipGram*

# List of Figures

# List of Tables

# List of Algorithms

# Chapter 1

# Introduction

## 1.1 Motivation

Recommender Systems (RSs) are systems or models with the goal of filtering items by identifying those with greater relevance to the user or client in an automatic and personalized manner. Considering how they work, they can generate fidelity, making the users spend more time on the platforms or increasing sales of a given service. Because of these effects, they are a crucial component in the daily routine of a significant part of the population, being present from streaming services and e-commerce platforms to social networks, thus making their use and presence indispensable. Due to this regular use, the RSs are constantly evolving, attracting new users and consequently generating more data, which allows further improvements to their functioning and creates a cycle of evolution.

Despite the first RSs being based on characteristics of the products, the so-called *“Content-based Filtering”*, their performance has been surpassed by RSs based on the consuming profile, identifying users similar to the one currently using the system, a process called *“Collaborative Filtering”* (CF). Other research has also proposed methods that use the products’ characteristics and the consuming history, generating the often called *“Hybrid RSs”*, but also referred to as RSs with *side information*.

Several authors (e.g., [44; 27; 68; 16]) proposed to investigate the possibility of performance improvement by using side information:

1. **User metadata**. Information regarding the user, such as a demographic profile. Com-

monly, this metadata contains the user's age group, occupation, average compensation, location, etc, and can be used to make the models learn similar consumer profiles.

2. **Item metadata**. Differently from the user metadata, the metadata of an item varies according to the domain of the application: music streaming platforms might have attributes such as genre, length, language, and others, while an e-commerce platform would use the category of the product, price range, and product brand as attributes.

3. **Context metadata**. More recently, researchers realized that the context in which a user consumes an item also impacts the recommendation that might be of interest. Therefore, platforms may recommend different items considering the time of the day, the user's current location, and the device used to access, among other attributes, to make a more specific recommendation [16].

Despite the research claiming performance improvement when side information is used, it is not always available, or the cost to obtain or use it is prohibitive. New strategies are necessary when only the user-item interactions are known.

The increase in available data combined with the algorithmic and hardware advances made the use of Neural Network- (NN-) based models feasible, especially those with a deep architecture, usually known as Deep Neural Networks (DNN), which constitute the sub-field of Deep Learning (DL) [13] under the Machine Learning (ML) field. In CV tasks, for example, the proposal of models such as VGG [54] and AlexNet [30], initially trained to perform a dummy task, such as image object classification, having a large image dataset as input, allowed the field to increase performance in a broad set of specific CV tasks since these models use the initial training to learn low-level, thus generic, image features, such as contours and shapes, which can be later used to improve the performance in a downstream task, which commonly has less available information. Similarly to CV, NLP models such as ELMo [40] and the ones that belong to the BERT [8] or GPT [43] families follow a similar strategy by pre-training the models for one or more generic task, such as text generation (i.e., next word prediction), and fine-tuning them for the desired task, for example, sentiment classification.

However, Transfer Learning cannot be easily applied to RSs; unlike CV and NLP, each application of RSs consists of a specific set of entities. For example, while any large image dataset can be used to learn contours and shapes, it is not possible, for example, to learn the representations of the users from a Netflix dataset and use them to improve performance in a Disney Plus dataset, since the users are not the same or, even if someone has accounts in both platforms, it may not possible to share their information due to privacy and data protection legislation. Recent research showed that when a company has multiple platforms for different services and can identify the user among them, a model can leverage information learned from a user regarding a service and use it to initialize the user representation when training a model for a different service, a process called by the authors as cross-domain pre-training [59].

Another barrier exists when thinking about item representation: it may not be reasonable to transfer item representation across different domains because there might not be any logical benefit from initializing a movie representation with the learned embedding of a song, for example. Despite this restriction, transferring item representations within the same domain is possible. However, it is not trivial and relies on the accuracy of text-matching (e.g., matching movie titles) algorithms.

The RSs branched through different paths, apart from pre-training and transfer learning. Some of these paths are:

1. **Model Robustness**. While many traditional recommender models relied on the user-item matrix (a.k.a. rating matrix or interaction matrix) or the interactions between users and items, possibly also using metadata, the popularization of NN made it easy to use them for recommendation. Examples of these models are the Neural Collaborative Filtering (NCF) [20], models built with autoencoders [34; 36], and transformer-based models [7; 56; 41].

2. **Semantic-Aware and Content-based models**. Considering that the CV and NLP areas were evolving faster, the RSs took advantage of these improvements to evolve the content-based recommenders through semantic awareness: the use of open data, textual, visual, and multimedia to create a component that better understands the available side-information [6]. Examples of this would be using an embedding-based NLP model to semantically understand the synopsis of a movie [55], or extracting visual features with a DNN to understand art preferences [66].

3. **Sequence- and Session-based recommendation**. Unlike the previous branches, these models structure the interaction data temporally to perform better in cases where the recent data has more impact than the general user history. The session-based scenario is a limitation on the Sequence-based where the system cannot identify the logged user but still benefits from promoting personalization based only on the current session information [56; 22; 23].

However, despite the reported advances, in 2019, two studies [10; 47] reviewed the reproducibility of papers in the RSs area, claiming that only a few methods were reproducible, and the ones that were frequently reported results in discrepancy with reality, outperforming baselines which were not trained and fine-tuned properly. These statements intensified the scrutiny upon new research, demanding better verification of results and conducting and executing reproducible experiments with open code and/or data.

## 1.2 Dissertation Statement

Challenged by the possibility and reported improvement of pre-training recommendation models with previously learned user representations [59], we hypothesize that overcoming the barriers of transferring item representations within the same domain may also benefit performance.

Furthermore, impacted by the recent studies regarding the performance improvement of having more robust models, especially in terms of the number of parameters [10; 47], we intend to evaluate our hypothesis over well-established models and datasets, which are often used as baselines, to understand the impact of the pre-training strategies for RSs.

This dissertation states that initializing item representation with the learned representations of the same items in different datasets or platforms may improve performance even when the models are as simple as they can be without using any side information.

The proposed statement leads us to pursue the following research questions, which guided the development of this work and are addressed in the next chapters:

**RQ1. Does pre-training improve a model's performance?** For this question, we evaluate the impact of using unsupervised and self-supervised techniques to generate embeddings, what we call non-task-specific pre-training, and the impact of supervised techniques, i.e., trained recommendation models to generate the embeddings, what we call task-specific pre-training.

**RQ2. How do out-of-vocabulary items impact a pre-trained model's predictive performance?** In this case, we investigate the task-specific pre-training further, understanding the impact of unknown items in the target dataset.

**RQ3. Does pre-training act as a model regularizer?** Previous research showed that using a more effective initialization strategy can lead to a regularization effect. In our case, we study if pre-training also acts as a regularizer.

**RQ4. How do the best results achieved with pre-training compare to the best-known results on the corresponding datasets?** Lastly, we intend to compare our best-performing models to the ones in the literature, understanding the different impacts depending on the addressed task.

## 1.3 Dissertation Contributions

The main contributions of this study are listed below:

1. Explore the idea of using same-domain public datasets to improve the performance of the models through pre-training and transfer learning without employing any side information;

2. Perform ablation experiments to understand the impact of the items not represented in the source dataset, often referred to as *out-of-vocabulary* (OOV) items;

### 1.3.1 Publications

Some of the results presented in this dissertation were earlier published as papers in conference proceedings as follows:

*UMAP'23* **Júlio B. G. Costa**, Leandro B. Marinho, Rodrygo L. T. Santos, and Denis Parra. 2023. Evaluating Pre-training Strategies for Collaborative Filtering. In Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization

(UMAP '23). Association for Computing Machinery, New York, NY, USA, 175–182. https://doi.org/10.1145/3565472.3592949

*IUI'22* Leandro Balby Marinho, **Júlio Barreto Guedes da Costa**, Denis Parra, and Rodrygo L. T. Santos. 2022. Similarity-based explanations meet Matrix Factorization via Structure-Preserving Embeddings. In 27th International Conference on Intelligent User Interfaces (IUI '22). Association for Computing Machinery, New York, NY, USA, 782–793. https://doi.org/10.1145/3490099.3511104

## 1.4 Outline

The subsequent chapters are organized as follows:

- **Chapter** 2 provides the background to understand the RSs and our study, also providing an overview of how the data is organized and represented, and portrays earlier research with similar methods to those used in our research, highlighting their differences.
- **Chapter** 3 defines the experimental setup and dataset choices, elaborates on the chosen pre-training approaches, and describes our evaluation scenarios.
- **Chapter** 4 further discusses the results and findings of our experiments over explicit and implicit feedback and cross-task transfer learning.
- Finally, we summarise our discoveries and contributions and indicate the following research in **Chapter** 5.

# Chapter 2

# Background and Related Work

In this chapter, we first review how data is represented for RSs and the principles of rating and ranking prediction. Second, we introduce the models we will use to evaluate our hypothesis. Third, we present the metrics used to assess performance. Next, transfer learning and pre-training will be discussed as well as the methods used in this research. Lastly, we discuss the related work.

## 2.1 Data Representation

The most simple RSs are applied over three different entities: (1) the users that interact with the system, denoted by $U = \{u_1, u_2, \ldots, u_N\}$; (2) the items available in the system, denoted by $I = \{i_1, i_2, \ldots, i_M\}$; and (3) the interaction of a user to an item, i.e. $r_{u,i}, \forall u \in U, \forall i \in I$.

This interaction $r_{u,i}$ is more frequently represented in two different ways: (1) *explicit feedback*, consisting in a discrete value that represents the rating given by the user to the item, usually in the $[1, 5]$ range; and (2) *implicit feedback*, when the user does not explicitly provide feedback, but the system can capture the interaction (e.g. the user bought an item or watched a video), usually represented with a binary value. Although less frequent, one might model an RS differently, such as capturing the implicit feedback and counting the times the user has interacted with that item.

In most RSs scenarios, the goal is the matrix-completion problem: the data is represented using a sparse $R_{N\times M}$ matrix such that each cell $r_{n,m}$ represents the interaction between the user $n$ with the item $m$. Figures 2.1a and 2.1b show an example of such representation for explicit and implicit feedback.

|  | $i_1$ | $i_2$ | $\dots$ | $i_{\|I\|}$ |
|---|---|---|---|---|
| $u_1$ | 1 | 3 | $\cdots$ | 4 |
| $u_2$ | $-$ | 4 | $\cdots$ | 5 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $u_{\|U\|}$ | 2 | $-$ | $\cdots$ | 3 |

(a) Explicit feedback matrix.

|  | $i_1$ | $i_2$ | $\dots$ | $i_{\|I\|}$ |
|---|---|---|---|---|
| $u_1$ | 1 | 1 | $\cdots$ | 1 |
| $u_2$ | 0 | 1 | $\cdots$ | 1 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $u_{\|U\|}$ | 1 | 0 | $\cdots$ | 1 |

(b) Implicit feedback matrix.

Figure 2.1: Interaction matrices. In the explicit feedback matrix, $-$ denotes the missing values.

The item prediction task, which uses implicit feedback, has gained popularity over the rating prediction task, which uses explicit feedback. This happened for several reasons, some of them are: (1) capturing interactions automatically allows the system to collect more data since it does not require an explicit user action to rate the interaction; (2) understanding user preferences more efficiently, considering that consumption is an indicator of preference by itself, and the interest might be increasing it; (3) each user has their line of thought for rating, and while one of them might rate something they disliked with 1 star, others might rate it with two stars; (4) explicit feedback can lead to bias since users might rate items they strongly liked or disliked and not do anything with all the others.

|  | $i_1$ | $i_2$ | $\dots$ | $i_{\|I\|}$ |
|---|---|---|---|---|
| $u_1$ | 0 | 0 | $\cdots$ | 1 |
| $u_2$ | 0 | 1 | $\cdots$ | 1 |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | $\vdots$ |
| $u_{\|U\|}$ | 0 | 0 | $\cdots$ | 0 |

Figure 2.2: Implicit feedback matrix with $t \geq 4$.

However, good rating prediction datasets are available, and researchers often binarize the rating matrix, transforming it into an interaction matrix. The most common transformation is to compare the given rating to a threshold $t$, such as $r'_{u,i} = r_{u,i} \geq t$, such that $r'$ is the resulting value of the cell in the implicit feedback matrix. Examples of this transformation can be seen from Figure 2.1a to Figure 2.1b when $t = 0$, and from Figure 2.1a to Figure 2.2 when $t = 4$.

Nevertheless, it is possible to question the efficacy of the user-item matrices as the data representation, considering the high number of missing values, making it usually called a "sparse matrix". Therefore, it is standard for authors to use and explore the sparsity measure, evaluated using Eq. 2.1, to justify performances and results obtained in their experiments. Also, with the increase in the number of users and items of a system, the interaction matrices tend to become even sparser with time, leading to the need for more memory, which may be a prohibitive factor when training recommendation models.

$$S(U, I, R) = 1 - \frac{|R|}{|U| \cdot |I|} \tag{2.1}$$

## 2.2 Recommendation Models

Since the proposal of factorization models [29] during the Netflix Challenge [2], it has become common to represent users and items using latent factors, i.e., continuous-valued vectors with a fixed $d$ number of dimensions, frequently referred to as *embeddings*. Using these embeddings, the matrices represent the users and items $P_{N \times d}$ and $Q_{M \times d}$, respectively, where $p_u$ is the embedding that represents user $u$ and $q_i$ represents item $i$.

### 2.2.1 Matrix Factorization (MF)

The goal of the MF model is to approximate the interactions between users and items $R$ by $\hat{R} = PQ^T$, such as the ratings can be calculated by $\hat{r}_{u,i} = p_u q_i^T$ for a given pair user-item [29; 47]. In this research, we will use a more recent version of MF, called *Biased MF*, that proposes the addition of biases, where $b_u$ is the bias of user $u$, $b_i$ is the bias of item $i$, and $\mu$ is a fixed global bias, defined by the average of all ratings in the training data. The ratings are predicted by Eq. 2.2.

$$\hat{r}_{u,i} = p_u \cdot q_i^T + b_u + b_i + \mu \tag{2.2}$$

The Biased MF aims to minimize the difference between the predicted and observed ratings. To do so, the squared error with L2 regularization, described in Eq. 2.3, is used as a loss function.

$$\sum_{u,i} (r_{u,i} - \hat{r}_{u,i})^2 + \lambda \left( b_u^2 + b_i^2 + \|p_u\|^2 + \|q_i\|^2 \right) \tag{2.3}$$

After evaluating the error, the representations and biases of users and items are updated iteratively through Stochastic Gradient Descent (SGD) according to Eq. 2.4, where $e$ is the prediction error and $\lambda$ and $\gamma$ are hyper-parameters that represent the learning rate and the regularization strength, respectively.

$$\begin{aligned} e &= \hat{r}_{u,i} - r_{u,i} \\ p_u &= p_u + \lambda(e - \gamma \cdot p_u) \\ q_i &= q_i + \lambda(e - \gamma \cdot q_i) \\ b_u &= b_u + \lambda(e - \gamma \cdot b_u) \\ b_i &= b_i + \lambda(e - \gamma \cdot b_i) \end{aligned} \tag{2.4}$$

### 2.2.2 Weighted Matrix Factorization (WMF)

Similarly to MF, the WMF [24; 38] model factorizes the user and item vectors to learn how to predict the preference $\rho_{u,i} = \{0, 1\}$ from user $u$ to item $i$, indicating if they liked (1) or disliked (0) that item, or whether they interacted with it (1) or not (0), for example. In this case, since there is no difference between users regarding how biased their ratings are, these terms are not present, and the preference is evaluated according to Eq. 2.5

$$\hat{\rho}_{u,i} = p_u \cdot q_i^T \tag{2.5}$$

As for MF, the goal of WMF is to minimize the error between the predicted preference $\hat{\rho}_{u,i}$ and the real preference $\rho_{u,i}$ and follows a very similar loss function, described in Eq. 2.6.

$$\sum_{u=1,i=1}^{N,M} c_{u,i}(\rho_{u,i} - \hat{\rho}_{u,i})^2 + \lambda \left( \sum_{u}^{N} \|p_u\|^2 + \sum_{i}^{M} \|q_i\|^2 \right) \quad (2.6)$$

However, instead of only using the user and item embeddings, it also proposes a confidence level $c_{u,i}$ of observing $\rho_{u,i}$, further described in Eq. 2.7, due to the possibility of unintended or misdirected consumption — for example, when a person buys something as a gift, or automatically plays the next song in a music platform.

$$c_{u,i} = 1 + \alpha \cdot r_{u,i} \quad (2.7)$$

In Eq. 2.7, $\alpha$ is a hyper-parameter, and $r_{u,i}$ is the explicit rating given by the user. Despite the model being well described through these equations, many mathematical optimizations are often implemented to improve training performance since, differently from MF, WMF needs to iterate through both the existing and non-existing interactions (i.e., $\forall u \in \{0, 1, \ldots, N\}, \forall i \in \{0, 1, \ldots, M\}$). These mathematical improvements are described in detail in the original paper [24].

### 2.2.3 Bayesian Personalized Ranking (BPR)

Similarly to the previous methods, BPR also works with embedding representations for both users and items. However, its key difference is that, instead of predicting a rating or the consumption of an item by a user, it works with pairwise comparisons between items [45; 11]. For example, given a user $u_k$, it will learn if their preference is for the item $i_a$ compared to $i_b$. As its name says, this model is based on the Bayesian assumption defined by the likelihood function in Eq. 2.8, where $\sigma$ is the sigmoid (i.e. the logistic) function $\sigma(x) = \frac{1}{1+e^{-x}}$.

$$p(i > j|u) = \sigma(\hat{x}_{u,i,j}) = \sigma(p_u \cdot q_i^T - p_u \cdot q_j^T) \quad (2.8)$$

The objective function of BPR is to maximize the likelihood of the observed preferences in the training data. Taking the negative log-likelihood, the objective function expressed by Eq. 2.9, where $I_u^+$ are the positive items for user $u$, $\Theta$ is the regularization term, and $\lambda$ controls the regularization strength.

$$BPR_{OPT} = -\sum_{u}^{U}\sum_{(i,j)}^{I_u^+} ln\sigma(\hat{x}_{u,i,j}) - \lambda\Theta \tag{2.9}$$

When structuring the data as input for this model, its performance will differ based on the definition of what the user has liked. Suppose only the consumption history is available (i.e. a binary matrix). In that case, it will learn to predict item consumption, which is interesting when applied to particular applications of RSs, such as e-commerce. A second possibility is predicting likeness, but this depends on the definition of what is liked: for example, if the user listens to a song A $10$ times and a song B $5$ times, we might understand that they prefer A over B. However, setting a likeness threshold is a common practice, as mentioned in Section 2.1, and causes the side effect of no distinction between consumed but not liked and not consumed items.

Considering these limitations of BPR, it may not perform well on some datasets due to the distribution of popularity of different items. Gantner et al. [11] proposed a modified version of BPR called WBPR, where its optimization criterion considers different weights for the negative items, as shown by Eq. 2.10,

$$BPR_{OPT} = -\sum_{u}^{U}\sum_{(i,j)}^{I_u^+} w_u w_i w_j ln\sigma(\hat{x}_{u,i,j}) - \lambda\Theta \tag{2.10}$$

where $w_u = \frac{1}{|I_u^+|}$, which balances the contribution of each user, and $w_i = 1$, uniformly weighting positive items.

BPR and WBPR use custom implementations of the stochastic gradient descent for optimization [45; 11].

## 2.3 Metrics

This section introduces the metrics used to measure performance in rating and ranking prediction tasks.

Our goal for the rating prediction task is to minimize the error between the prediction and the given rating. Therefore, although the users can only assign a discrete rating (e.g., from $1$ to $5$ stars), the problem can be seen as a regression since the ratings are a value to be predicted. Thus, we use the Root Mean Squared Error (RMSE), presented in Eq. 2.11

as the primary performance metric, where $r_{u,i}$ is the rating given from user $u$ to item $i$ and $\hat{r}_{u,i}$ is the predicted rating between the same pair, and $D$ is the dataset composed of $(u, i, r)$ triplets.

$$RMSE = \sqrt{\frac{1}{|D|} \sum_{u,i,r}^{|D|} (r_{u,i} - \hat{r}_{u,i})^2} \tag{2.11}$$

Regarding the ranking prediction tasks, it is necessary to evaluate whether the item is recommended and the ranking of the item among the recommended items. For this reason, it is expected to present the metrics with the suffix $@K$, where $K$ represents the number of recommended items. We use the following set of metrics:

1. Precision$@K$ measures the number of items relevant to the user concerning the number of items predicted. It is defined by Eq. 2.12, where $Rel_u$ is the set of relevant items for user $u$ and $Rec(u, k)$ is the set of $k$ recommended items for user $u$.

$$\text{Precision}(u, k) = \frac{Rel_u \cap Rec(u, k)}{k} \tag{2.12}$$

2. Recall$@K$ measures the number of items predicted that are relevant to the user in relation to the number of relevant items for that user. It is defined by Eq. 2.13, where $Rel_u$ is the set of relevant items for user $u$, $|Rel_u|$ is the number of items relevant to $u$, and $Rec(u, k)$ is the set of $k$ recommended items for user $u$.

$$\text{Recall}(u, k) = \frac{Rel_u \cap Rec(u, k)}{|Rel_u|} \tag{2.13}$$

3. NDCG$@K$ is an acronym for Normalized Discounted Cumulative Gain, a ranking measure that considers both the prediction's relevance and its ranking. It is defined by Eq. 2.15, where $Rel_u$ are the relevant items for user $u$, and $rel(u, i)$ is the function defined by Eq. 2.14

$$rel(u, i) = \begin{cases} 1 & \text{if } i \in Rel_u \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

$$
\begin{aligned}
\text{DCG}(u,k) &= \sum_{i}^{k} \frac{2^{rel(u,i)}-1}{log_2(i+1)} \\
\text{IDCG}(u,k) &= \sum_{i}^{|Rel_u|} \frac{2^{Rel_{ui}}-1}{log_2(i+1)} \\
\text{NDCG}(u,k) &= \frac{DCG(u,k)}{IDCG(u,k)}
\end{aligned}
\tag{2.15}
$$

## 2.4 Pre-training and Transfer Learning

In this section, we explore the concepts of pre-training, transfer learning, and methods that generate latent factors.

### 2.4.1 Transfer Learning

Traditional machine learning is characterized by training models with a dataset and evaluating it with a separate testing dataset. However, more complex problems often require more training data, which is not usually available. A second requirement is that the training and testing data should have the same input feature space and distribution since different data distributions may lead to degradation in the learning process [60].

Transfer learning is used to improve the performance of a target model by leveraging the knowledge of a source model trained with information from a related domain. An interesting analogy is thinking about learning how to play piano: someone who already plays guitar will likely learn faster than someone with no musical background [39].

VGG [54], a DNN proposed for CV tasks, is one of the best examples of transfer learning: after training the model using the large ImageNet dataset, one can freeze the weights of the convolutional and pooling layers, which have already learned to extract visual features from images and learn or fine-tune the weights of the fully connected layers to perform a related task, e.g., animal species classification. Figure 2.3 shows the architecture of the VGG model. The convolutional and pooling layers are shown in yellow and orange, while the fully connected layers are pink and purple.

Despite the performance improvements, transfer learning may also lead the model to mistakes depending on the data used to train and evaluate the model, and practitioners and

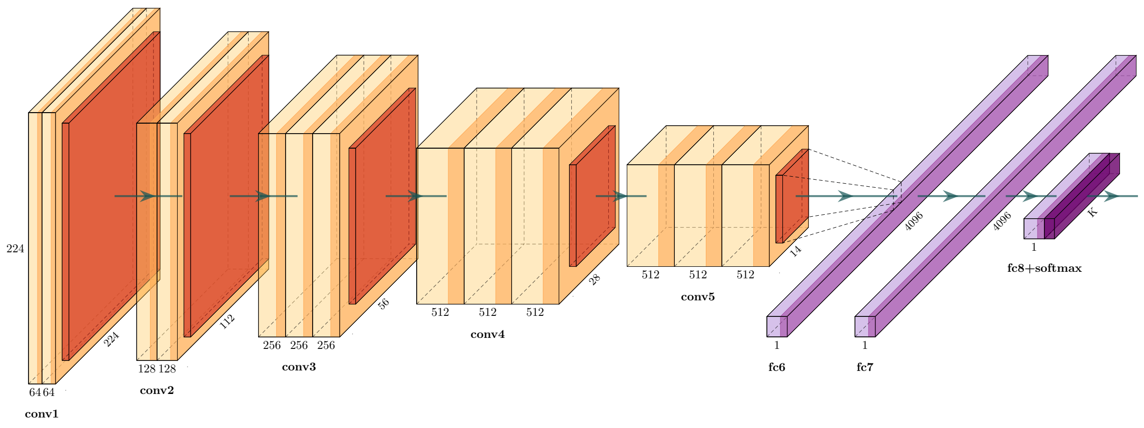

Figure 2.3: VGG Network

researchers must evaluate the results thoroughly [48].

## 2.4.2 Pre-training

As in transfer learning, pre-training aims to improve a model's performance by leveraging knowledge from a large dataset. However, while transfer learning uses the features learned as byproducts of a trained model, pre-training often uses unsupervised or self-supervised learning methods, capturing general patterns of features present in the data [37].

One of the best examples of pre-training is the BERT [8] model, which performs self-supervised training for the Masked Language Modelling (MLM) and Next Sentence Prediction (NSP) tasks before fine-tuning the learned representations for the desired task (e.g., question answering).

The critical differences between transfer learning and pre-training are their objectives and training data. Transfer learning prioritizes enhancing performance in a singular task, while pre-training aims to acquire generic representations beneficial across numerous tasks. Regarding training data, transfer learning requires a substantial dataset relevant to the source task for training the initial model. Contrastively, pre-training can directly use the dataset to learn good initial representations, later adjusted to the downstream task in the fine-tuning stage.

In summary, pre-training is a step before transfer learning, where the model learns generic features or representations before being adapted to a specific task through transfer learning. Both techniques are essential in leveraging large datasets and improving model performance, especially in scenarios with limited labeled data.

### 2.4.3 Latent Factor Generation

Here, we present two methods used in this research to generate latent factors.

#### Principal Component Analysis (PCA)

Principal Component Analysis (PCA) [1] is a technique that analyzes a dataset described by several dependent variables which are, in general, inter-correlated. The main goals of PCA are (1) extracting the most essential information from the data and (2) compressing the size of the data by keeping only the most relevant information. To do so, it performs a sequence of mathematical operations to understand which features better represent the variance contained in the data:

1. **Centering the data**. Initially, PCA computes the mean vector $\overline{x}$ and centers each feature vector based on the mean: $x_i' = x_i - \overline{x}$.

2. **Computing the covariance matrix**. Having the $X\prime$ matrix of centered data, calculate its covariance matrix: $E = \frac{1}{n}\sum_{i=1}^{n} x_i' x_i'^T$.

3. **Eigenvalue Decomposition**. This is the most important step of PCA. From the covariance matrix, we retrieve the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ from the equation $det(E - \lambda I) = 0$, where $I$ is the identity matrix. With the eigenvalues at hand, the next step is to solve the equation $(E - \lambda I)V = 0$ to find the eigenvectors $v_1, v_2, \ldots, v_n$, called *principal components*.

4. **Selecting the principal components**. Now that the principal components are found, we sort them by their corresponding eigenvalues in decreasing order since they represent the variance contained in each vector, creating a matrix $V = \{v_i, v_j, \ldots, v_n\}$ by stacking the $d$ most important components, where $d$ is the desired number of components given as input.

5. **Projection**. Lastly, multiply the centered data matrix $X'$ by the selected $V$ eigenvectors: $Y = X'V$.

Considering how it works, PCA can only be applied to continuous-valued numerical data as a dimensionality reduction technique.

### Word Embeddings

Proposed for NLP, Word Embeddings are techniques to generate numerical representations for words, including semantic features. One of the first works by Mikolov et al. [37] proposed the Skip-Gram (SG) and the Continuous Bag Of Words (CBOW) models, which extract the semantics of the words based on a $w$-sized window of tokens or words. Figure 2.4 shows the architecture of both models.

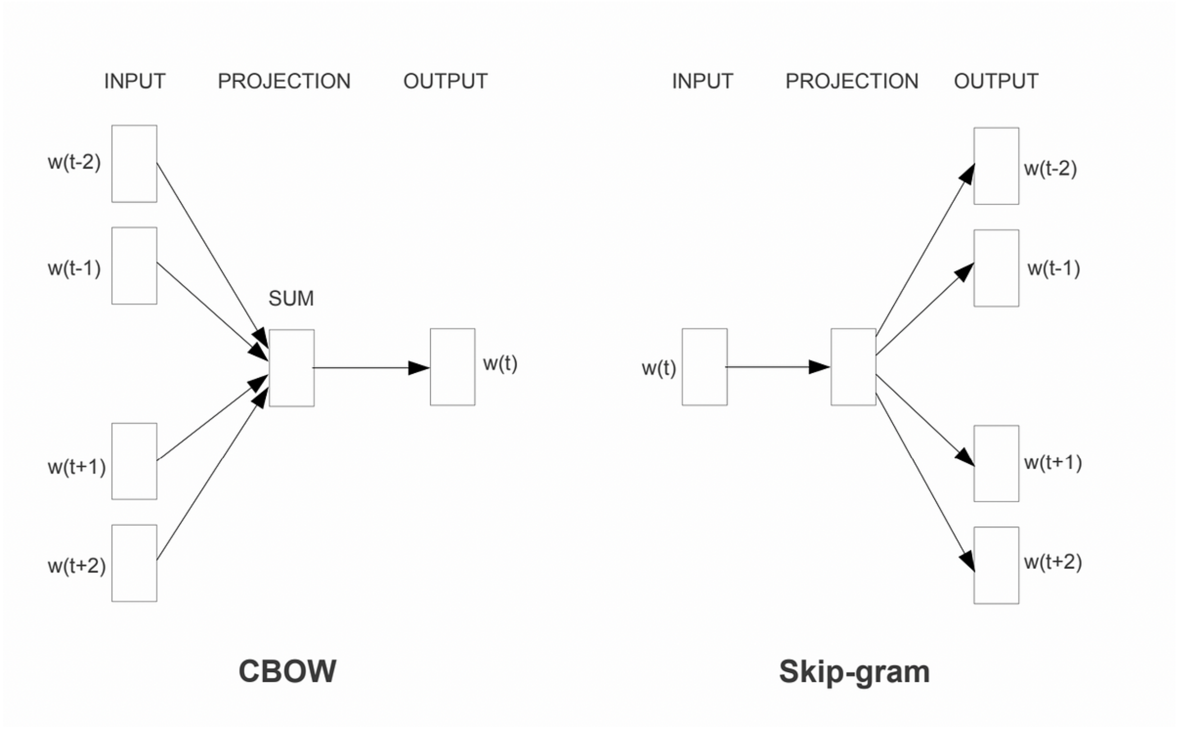

Figure 2.4: CBOW and Skip-Gram model architectures.

Consider a size window $w = 5$ and the corpus $C$ of known tokens. SG and CBOW randomly initialize each token's $d$-sized embedding representations. The SG model is trained by having the word $w(t)$ as input and trained to predict the two tokens before $(w(t-2)$ and $w(t-1))$, and the two tokens after it $(w(t+1)$ and $w(t+2))$. CBOW performs the opposite operation: given the two tokens before $(w(t-2)$ and $w(t-1))$ and the two tokens after $(w(t+1)$ and $w(t+2))$, predict the current token $w(t)$. Although both models can capture the semantics of tokens, they are limited in cases where the same word can have multiple meanings. For example, the word *bat* can be used to represent the mammal and the sports equipment used in baseball, and the word *ring* can mean a piece of jewelry or be used as a verb indicating a sound (e.g., "Your phone is ringing").

## 2.5 Related Work

In the last decade, the advances achieved with AlexNet [30], VGG [54], and other models trained with the ImageNet dataset [49] made transfer learning the standard practice for CV tasks [52]. The same goes for the usage of embeddings in NLP tasks, which started with self-supervised strategies such as SkipGram and CBOW [37] and evolved into the complex training strategies used in the BERT [8] and GPT [43] models. These advances allowed the research and industry communities to have model libraries, such as HuggingFace [63], which will enable models to be used off-the-shelf or further adjusted with additional training using a specific dataset to perform a similar task, a process often called *fine tuning* [13; 42].

Considering the challenges associated with applying transfer learning in the context of RSs, researchers noticed that an effective (i.e., non-random) initialization strategy could expedite the learning process, enhance accuracy, and yield more transferable representations for the model [12; 19]. To illustrate, Seuret et al. [51] demonstrated that networks initialized with Principal Component Analysis (PCA) resulted in accelerated and more stable training. Additionally, these networks outperform their counterparts with random initialization, particularly in tasks related to document processing. However, despite its connection with learning user and item embeddings in the context of MF for CF, research in this domain remains scarce.

With this empirical support that PCA initialization leads to better performance, SimFactor [21] is likely one of the first works that applies it to MF, using the user/item metadata matrices as input and employing the resulting vectors as the initial MF weights. Our goal, however, is to evaluate performance improvements when no side information is available.

Another approach is the initialization of the user/item embeddings with a self-supervised method such as the early Word Embedding models, in particular, the SkipGram (SG) and CBOW strategies [37] for the item embeddings and Doc2Vec for user embeddings [31].

The usage of these techniques for RSs was already explored: Grbovic et al. [14] introduce *prod2vec*, a method that consists of applying SG to the sequence of bought products, and *bagged prod2vec*, when users buy more than one item in a single purchase; Barkan and Koenigstein [5] proposed *item2vec*, which uses a Skip-Gram (SG) with Negative sampling for generating item embeddings when the user cannot be identified — a scenario similar to Session-based recommendation — due to its nature of preserving item similarities. Liang et al. [35] introduced *CoFactor*, a model with an MF component that is regularized by an Item Embedding co-occurrence component. Although these studies use Word Embedding models — or parts of them —, none evaluated initializing the MF item embeddings with the retrieved representations applying further training.

When delving into applications of non-random initialization for MF, Ar [3] applies a custom initialization to a Probabilistic MF based on the distribution statistics of the datasets; Han et al. [15] introduce *GLocal-K*, a kernel-based matrix completion method by pre-training and fine-tuning the data with local and global kernelized representations of the ratings matrix. Lastly, Wang et al. [59] addressed the cross-domain recommendation problem by learning the user representations in an initial domain and using them to initialize the embeddings on the target domain nodes through a contrastive self-supervised graph NN.

# Chapter 3

# Method

In this chapter, we define our methodology, elaborate on our dataset choices, detail the chosen pre-training approaches, and explain how the transfer learning was performed and how the data was processed.

## 3.1 Datasets

In other application areas of ML, pre-training and transfer learning for recommendation can be performed for different contexts without major concerns: for example, in NLP, a model can be pre-trained using text from Wikipedia and fine-tuned to classify the emotion (e.g., anger, disgust, fear) of a tweet, since the data is not associated with any entities other than words.

The entities present in RSs are usually users and items. The same entities must be present in both datasets to transfer their learned representations fully. Since this is a significantly restricted scenario, researchers proposed transferring the user's representations, the Cross Domain scenario [59] despite the difficulties of identifying the same user's presence in datasets of different sources and, even when they have the same source, there might be difficulties due to regulations on using personal information [9].

Considering these constraints in transferring user representations, we propose only transferring the items' representations with the restriction of doing so within the same domain. Due to the high availability of public datasets, we chose to perform our experiments within the context of movie recommendations, using the MovieLens and Netflix Prize datasets.

The MovieLens [17] is a dataset collected by GroupLens Research[1], publicly available. Since its first version, which had 943 users, 1682 movies, and a total of 100k ratings, GroupLens released other 4 other versions of the dataset, containing 1M, 10M, 20M, and 25M ratings, this latter with 162,541 users and 62,423 movies, with interactions between January 09, 1995, and November 21, 2019. The statistical description of the datasets is shown in Table 3.1.

On the other hand, the Netflix Prize [2] dataset was first released on October 2, 2006, with 25M ratings. However, the competition lasted until 2009, and each year, Netflix released an extension of the dataset with approximately another 25M ratings, totaling roughly 100M ratings, with 480,189 users and 17,770 movies. Although Netflix no longer has a page for the challenge, the dataset is publicly available at Kaggle[2]. The statistical description is also shown in Table 3.1.

Table 3.1: Statistical description of the datasets.

| Dataset | $\lvert U\rvert$ | $\lvert I\rvert$ | Interactions | Sparsity |
|---|---|---|---|---|
| ML100k | 943 | 1682 | 100,000 | 94.85% |
| ML1M | 6040 | 3706 | 1,000,209 | 95.53% |
| ML10M | 69,878 | 10,677 | 10,000,054 | 98.65% |
| ML20M | 138,493 | 26,744 | 20,000,263 | 99.46% |
| ML25M | 162,541 | 59,047 | 25,000,095 | 99.73% |
| Netflix | 480,189 | 17,770 | 100,480,507 | 98.82% |

## 3.2 Pre-Training

This section describes and justifies the pre-training approaches applied in this research.

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

[2] https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data

### 3.2.1 Non-task-specific

Principal Component Analysis (PCA) [1] is an *unsupervised* method frequently used for dimensionality reduction. It takes a matrix as input and performs a series of mathematical operations to reduce it to a second matrix with the same number of rows and a given $d$ number of columns (dimensions) while conserving the most relevant information.

Earlier applied in related research [51; 21], one of our studies showed that initializing user or item embeddings with PCA not only improves the predictive performance of the models but also presents a tradeoff between the performance and the explainability of the recommendations, since it preserves the original neighborhoods of users or items.

In this work, we use PCA as a non-task-specific pre-training, due to its unsupervised nature, to initialize the item embeddings using the original sparse rating matrix $R \in \mathbb{R}^{|U|\times|I|}$ as input, such that the item embeddings are defined by $\mathbf{Q} = PCA(\mathbf{R}^T, d)$.

A second strategy for non-task-specific pre-training is using self-supervised methods. Although there are many self-supervised techniques able to generate embedding representations, Word2Vec [37] revolutionized NLP by providing a method that produces general-purpose embeddings by considering the context of the words. Although more complex methods exist to generate word embeddings, such as transformer-based ones, Word2vec is still widely used due to its simplicity and efficiency.

In this work, we adopt the Skip-Gram (SG) algorithm, one of the Word2Vec implementations, which is a shallow NN trained by receiving a word as input and trying to predict a $w$-sized window of its surrounding words (i.e., its context), as described in Section 2.4.3. In NLP, Skip-Gram would structure a sentence as a sequence of tokens, so that it can identify the surrounding tokens as its context. For example, the token representing the word *is* can be used to predict the tokens {*The*, *sky*, *the*, *limit*} in the sentence *"The sky is the limit"*.

We use SG to generate the item representation based on the user's consumption history and use them to initialize the $\mathbf{Q}$ matrix of item embeddings in MF. This embedding generation process is similar to the one executed by Barkan and Koenigstein [5]. However, they only used the learned representations to predict, similar to the Next Sentence Prediction (NSP) task in NLP, while we use them as initial item-embedding representations in a recommendation model. Table 3.2 shows applications of Skip-Gram for NLP and recommendation.

Table 3.2: Application of Skip-Gram for NLP and pre-training item representations, using a $w = 5$ sized window, showing the token and movies, and their respective IDs

| | **Input** | **Output** |
|---|---|---|
| NLP | “curly” | {“She”, “has”, “brown”, “hair”} |
| | $w_3$ | $\{w_1, w_2, w_4, w_5\}$ |
| Recommendation | *Monsters, Inc* | {*Jumanji*, *Tarzan*, *Toy Story*, *Tangled*} |
| | $i_{127}$ | $\{i_{50}, i_{82}, i_{174}, i_4\}$ |

Although we are grouping PCA and Word2Vec as non-task-specific pre-training, a critical difference between them worth mentioning is that, while PCA considers the value of the rating given by the user to the item when evaluating similarity, Word2Vec only considers the co-occurrence of items in the context window, even if the ratings are contrasting.

### 3.2.2 Task-specific

The previous approaches provide item embeddings as byproducts of tasks related to discovering or preserving hidden local structures. Still, we can also generate embeddings as byproducts of a recommendation model itself. This approach is more similar to transfer learning, where a model is trained using a larger dataset, also called *source* model and dataset, and used to initialize the weights of a second model trained on a smaller dataset with a more specific goal, also called *target* model and dataset. In our case, we can train an MF model using a large dataset and use the learned item representations as initialization for a second MF model, which will be trained with the dataset we want to recommend. It is also important to realize that the source model can also use an unsupervised or self-supervised initialization, which allows a broader number of pre-training possibilities.

## 3.3 Experimental Setup

This section details how we prepared the datasets and performed the transfer learning.

### 3.3.1 Data Processing

The transfer learning process is frequently applied to disjoint datasets for two main reasons: (1) the data available for the desired task is limited and not sufficient to train a robust model, and (2) if the source and target datasets have common data, the data used to train the source model might be present in the data used to evaluate the performance of the target model, which characterizes a *data leakage* problem. Therefore, to properly evaluate the impact of these techniques for recommendation, we need disjoint datasets.

Finding the same item in different datasets is not trivial. To perform our studies, we conducted two sets of experiments: the first consisted of using the largest MovieLens dataset to create two disjoint datasets, one larger dataset acting as source and one smaller acting as target; the second consisted of using the Netflix dataset as source, and the MovieLens dataset as target. Although both datasets are in the movies domain, it is essential to mention that they are different systems: Netflix was, initially, a movie-renting platform, while MovieLens is a movie rating and recommendation website.

This first experiment allowed us to match items across the source and target datasets by their ID. However, when generating the disjoint datasets from the MovieLens 25M dataset, the largest and most recent version, we had to respect some definitions of datasets during the transfer learning process:

1. The MovieLens datasets are frequently called in the literature as *dense* or *core* datasets, which are datasets where each user has at least a $c$ number of interactions, usually being $c = 10$, often also extended to the items.

2. Since we cannot identify the same person as a user of different platforms, we should assume they are different users.

The first definition is essential to avoid the problem known as *cold-start* of recommending unknown or poorly known items to a user or known items to a new or recently created user. This problem exists in real recommendations but is addressed differently, often combining collaborative and content-based models or using different strategies, such as the session-based recommendation, where the models are trained using only item representations and can handle unknown users.

Considering these definitions, the source and target datasets should remain core datasets, and no user should be present in both. To do so, we created a simple script described by Algorithm 1 in Appendix A.

### 3.3.2 Item Matching

The second set of experiments is more tricky since we need to match items across different datasets with no common ID. To perform this operation, we first needed to match items across datasets. However, the only information we can access is the movie's title and year of release since we are not using metadata.

To perform the matching between the items, we followed a 3-stage procedure:

1. We first pre-process the movies' titles to lowercase and the same encoding. This is important because there are movies in different languages, and the casing is just a detail. Next, we try to match using the processed title and year of release;

2. With the still unmatched items from the previous stage, we try to match using only the title. This stage allows matching movies where the year of release might be incorrect in one of the datasets;

3. Lastly, for the remaining unmatched items, we apply a weighted set of *string matching* algorithms, matching an item of the source dataset with the higher-scored item in the target dataset when the score reaches an acceptance threshold.

For the third stage, we used the available implementation of Indel [65] and Levenshtein [33] similarities in the *Python Levenshtein* library[3] [4] and the cosine similarity to score pairs of item titles. We also evaluated the Jaro [18; 26] and Jaro-Winkler [61; 62] matching algorithms, but the results were inferior. While the cosine is a token-based similarity measure, the Indel and Levenshtein are character-based, and setting a higher weight to the cosine similarity resulted in better matches. Finally, we normalized the similarities to the $[0, 1]$ scale. We retrieved the item with the higher similarity, only considering it a good match if the average weighted similarity surpassed the $75\%$ acceptance threshold.

[3] `https://github.com/maxbachmann/Levenshtein`

The first stage should find most matches correctly, while the second and third are fallbacks that may grant additional matches. Still, there are no guarantees that these matches are accurate. For example, "*A Star is Born* (2018)" is a remake of "*A Star is Born* (1954)", and ignoring the year of release might cause a mismatch. However, it is not guaranteed that all items in the target dataset will be matched with one in the source dataset. This scenario is similar to the *Out of Vocabulary* (OOV) one in NLP, and the embeddings of the unmatched items should be initialized randomly or following a different strategy.

### 3.3.3 Evaluation Scenarios

Using the chosen pre-training strategies, we define two scenarios: *Target2Target* and *Source2Target*.

Figure 3.1 shows the *Target2Target* scenario. We use the non-task-specific approaches to generate the item embeddings, followed by model training (or fine-tuning) on the same dataset. It is crucial to mention that since the same dataset is used for pre-training, fine-tuning, and performance evaluation, only the train split is used as input for the pre-training strategy and fine-tuning, while a separate unseen partition is used for testing.

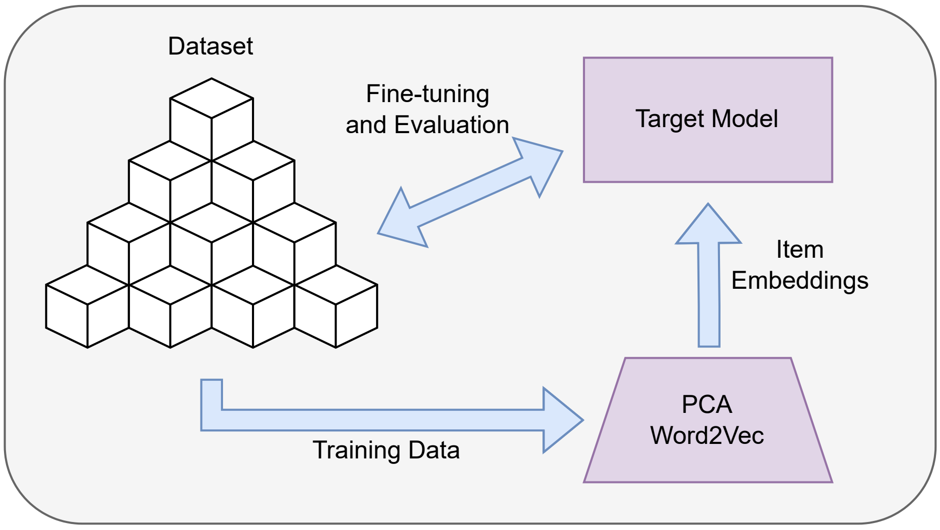

Figure 3.1: Target2Target scenario

On the other hand, the *Source2Target* uses a source dataset to train a recommendation model. After learning the item representations, embeddings corresponding to items shared between the source and target datasets are transferred to a second model, which will use the target dataset for training (or fine-tuning) and evaluation. This scenario is shown in Figure 3.2.

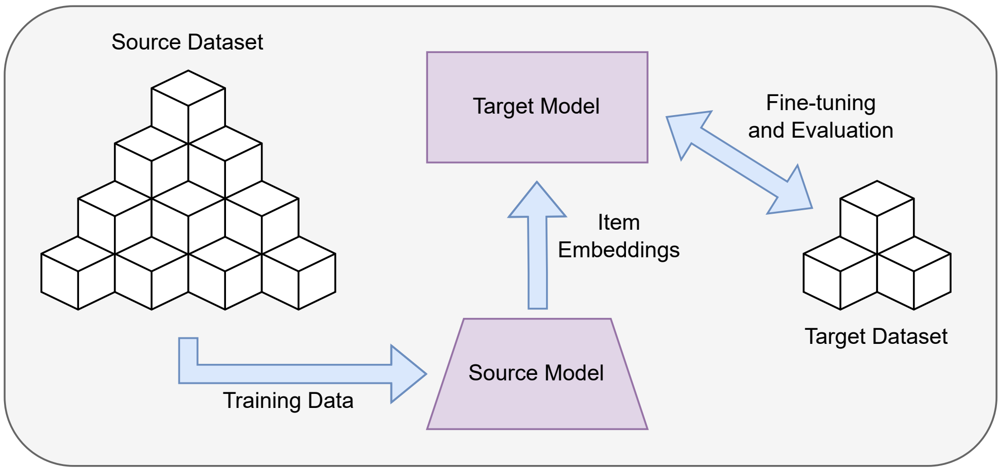

Figure 3.2: Source2Target scenario

We summarise both scenarios in Fig. 3.3, where the yellow arrow indicates a step unique to the *Target2Target* scenario, the blue arrow shows a step unique to the *Source2Target* scenario, and the red arrows indicate common steps for both. It is worth mentioning that the embedding generator can perform unsupervised or self-supervised (from either the source or target datasets) and supervised pre-training (from the source dataset only).

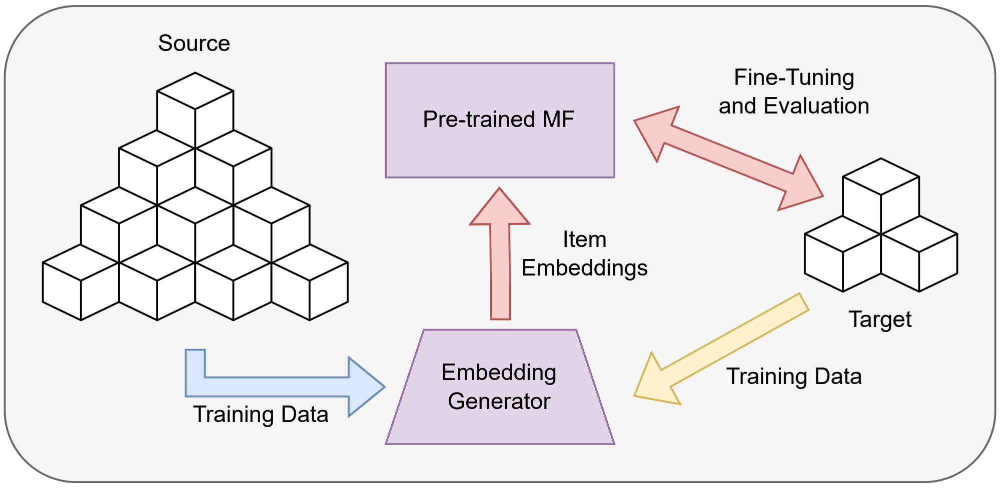

Figure 3.3: Evaluation Scenarios

Inspired by how transfer learning works, by leveraging information trained in a related task, and considering that our experiments iterate in both Rating and Ranking prediction, we also propose studying the transfer learning between tasks: *Ranking2Rating* and *Rating2Ranking*. Still, both cases follow the *Source2Target* pipeline: item embeddings from a source model trained on a large dataset for ranking prediction will be used to pre-train a rating prediction model, and vice-versa. Figure 3.4 depicts the *Rating2Ranking* case as an example.

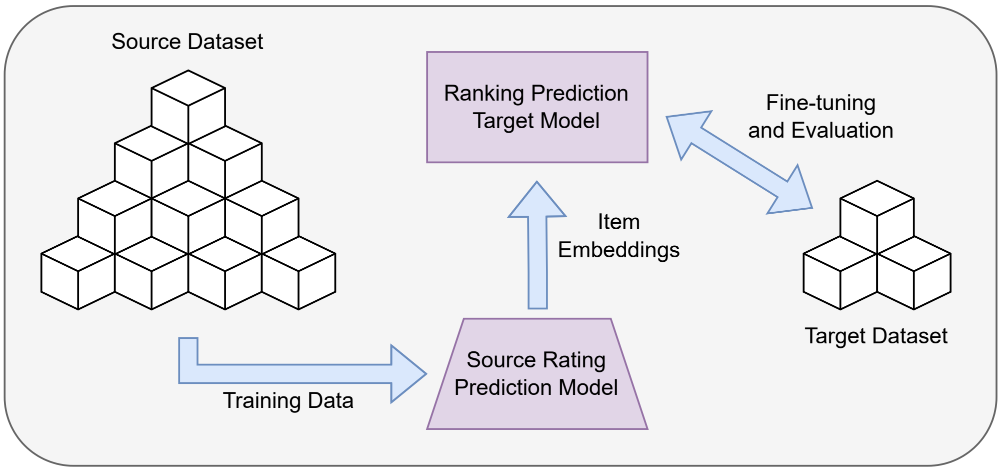
Source Dataset
Ranking Prediction Target Model
Fine-tuning and Evaluation
Item Embeddings
Target Dataset
Source Rating Prediction Model
Training Data

Figure 3.4: *Rating2Ranking* case in the *Source2Target* scenario

# Chapter 4

# Evaluation

This chapter elaborates on our discoveries, addresses the proposed research questions, and analyzes the outcomes of our experimental investigations over explicit feedback, implicit feedback, and cross-task transfer learning, respectively.

## 4.1 Explicit Feedback

In this section, we elucidate the particulars and outcomes of our investigation of the rating prediction task utilizing the biased version of the Matrix Factorization model. The chosen hyperparameters include $d = 128$ and the number of epochs $e = 100$, with $\mathbf{P}$ and $\mathbf{Q}$ following a normal distribution $\mathcal{N}(0, 0.1)$, a learning rate $\alpha$ set at 0.001, and a regularization parameter $\lambda$ established at 0.04. This configuration, documented in [47], has been reported as optimal for Biased Matrix Factorization in MovieLens 10M. Considering that we adopted similar datasets, using the same hyperparameter values makes sense. They worked well across our experiments, considering pre-trained and randomly initialized models.

We adapted the Biased MF Cython implementation publicly available in the *Surprise* [25] v1.1.1 Python library[1] to perform the rating prediction experiments.

[1] `http://surpriselib.com/`

**RQ1. Does pre-training (task- or non-task-specific) improve a model's predictive performance?**

To answer this question, we first analyze the *Target2Target* scenario, which includes unsupervised and self-supervised pre-training. In this case, we are evaluating the impact of pre-training on all the MovieLens datasets. We perform a 5-fold cross-validation, generating random training (80%) and testing (20%) splits, and report the average Root Mean Squared Error (RMSE) over the test folds in all evaluation scenarios.

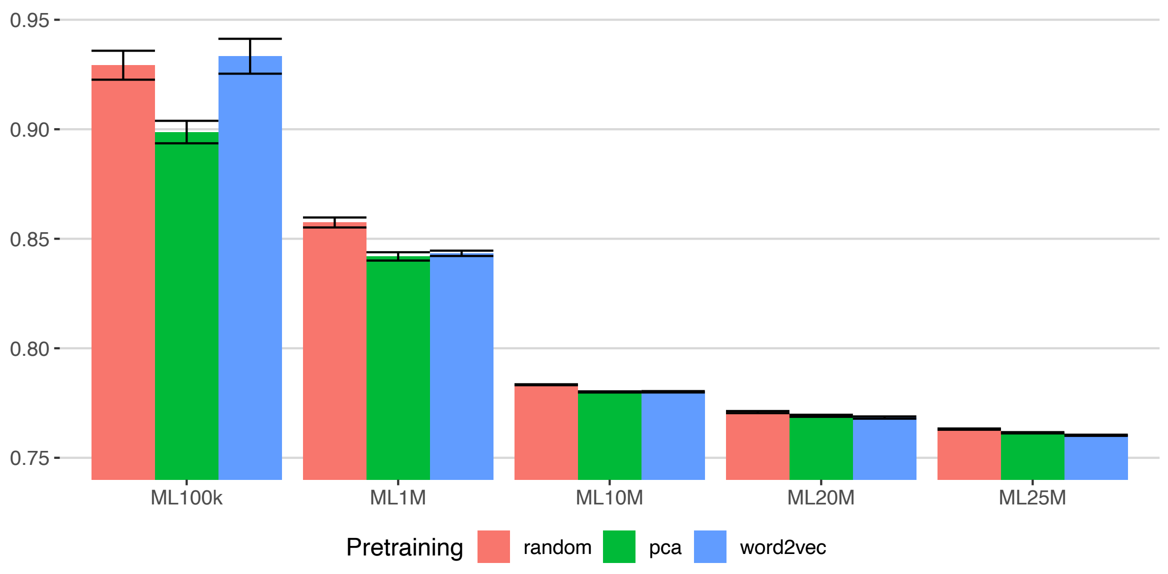

Figure 4.1: Predictive performance on the *Target2Target* pre-training scenario. Error bars denote the standard deviation from 5-fold cross-validation.

Table 4.1 depicts the basic statistics of the source and target datasets used to evaluate the *Source2Target* scenario. Figure 4.1 illustrates the outcomes expressed regarding RMSE, with lower values being preferable. Except for MovieLens 100k, where Matrix Factorization (MF) pre-trained with Word2Vec fails to surpass the baseline initialized randomly, both PCA and Word2Vec pre-training consistently demonstrate superior performance across all other datasets, particularly in the cases of MovieLens 100k and 1M. Noticeably, pre-training's efficacy diminishes as dataset sizes increase, as expected.

Figure 4.2 illustrates the outcomes of the *Source2Target* scenario, employing MovieLens 24.9M and MovieLens 24M as source datasets, along with their respective 100k and 1M counterpart sampled target datasets in the supervised pre-training setting. Results for

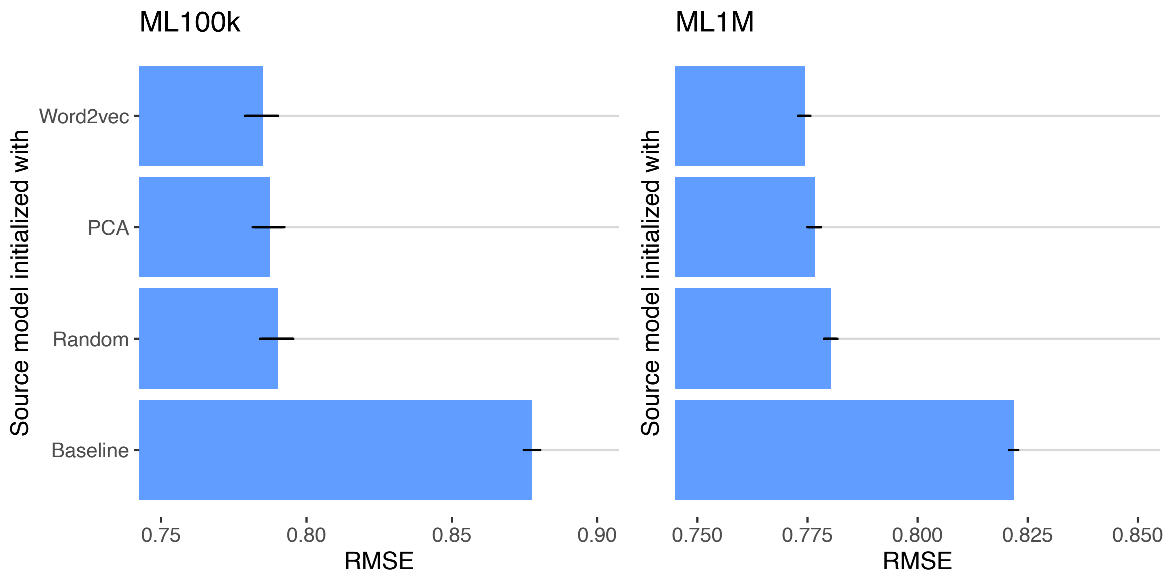

Figure 4.2: Predictive performance on the *Source2Target* pre-training scenario. Error bars denote the standard deviation from 5-fold cross-validation.

various pre-training strategies are presented next. For example, the *Word2Vec* approach initiated the source MF model with Word2Vec, followed by fine-tuning in the target dataset. We noticed that all strategies yielded a significant benefit compared to the baseline, especially Word2Vec. Hence, we answer RQ1 positively, especially regarding the *Source2Target* scenario. Appendix B comprises the results of further experiments in the *Source2Target* scenario, where we performed the same experiments sampling source and target datasets from the MovieLens 10M and 20M versions.

However, one thing to notice is that even these results may also present OOV items, meaning the embeddings not found in the source model were randomly initialized. We raised the following research question to understand better these cases, which are closer to pre-training in real-world datasets.

**RQ2. How do out-of-vocabulary items impact a pre-trained model's predictive performance?**

To answer this question, we restricted the number of embeddings transferred by randomly sampling a proportion of the embeddings considering the proportion levels $[0, 0.1, 0.2, \ldots, 0.9, 1]$, $0$ meaning no transfer, and $1$ meaning complete transfer. We also considered three variants for initializing the item representations not found in the source model and the source models: random initialization, PCA, and Word2Vec. This results in a total of nine settings evaluated.

Table 4.1: Basic statistics of the datasets generated to evaluate the *Source2Target* strategy.

| **Dataset** | **Source** | | | | **Target** | | | |
|---|---|---|---|---|---|---|---|---|
| | $\|U\|$ | $\|I\|$ | $\|R\|$ | **Sparsity** | $\|U\|$ | $\|I\|$ | $\|R\|$ | **Sparsity** |
| **MovieLens 10M** | 69,209 | 10,648 | $\approx$ 9.9M | 98.92% | 669 | 6530 | $\approx$ 100K | 98.16% |
| | 62,773 | 10,626 | $\approx$ 9M | 98.92% | 7105 | 9800 | $\approx$ 1M | 98.85% |
| **MovieLens 20M** | 137,783 | 25,838 | $\approx$ 19.9M | 99.55% | 710 | 7644 | $\approx$ 100K | 98.52% |
| | 131,773 | 25,605 | $\approx$ 19M | 99.54% | 6720 | 14,467 | $\approx$ 1M | 99.17% |
| **MovieLens 25M** | 161,938 | 56,608 | $\approx$ 24.9M | 99.78% | 603 | 9053 | $\approx$ 100K | 98.52% |
| | 156,100 | 56,302 | $\approx$ 24M | 99.78% | 6441 | 22,877 | $\approx$ 1M | 99.45% |
| **Netflix** | 455,525 | 1591 | $\approx$ 22.3M | 97.53% | 943 | 1682 | ML100K | 93.69% |
| | 467,479 | 3295 | $\approx$ 43.1M | 97.75% | 6040 | 3683 | ML 1M | 96.40% |

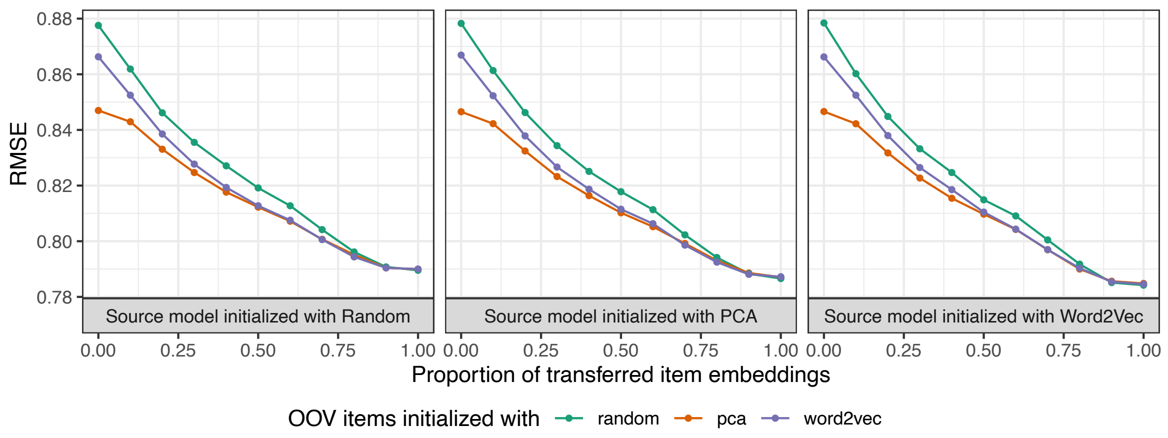

Figure 4.3: Predictive performance on MovieLens 100k using the *Source2Target* strategy considering MovieLens 24.9M as the source.

Figure 4.3 shows the results of these settings on the MovieLens 24.9M/MovieLens 100k source/target pair. The colors indicate the OOV initialization in the target dataset, while each facet indicates the initialization in the source dataset. In all cases, we can see that the RMSE decreases monotonically as more embeddings are transferred from source to target. In the best case, the RMSE decreases from 0.87 from the randomly initialized model with no transfer (baseline) to $\approx 0.78$ from a model pre-trained with embeddings generated from a source model initialized with Word2Vec.

We also evaluated the impact of pre-training considering a larger source dataset, with $\approx$ 1M ratings. Figure 4.4 shows the MovieLens 24M/1M source/target pair results. We can see that the impact of pre-training is similar to the previous example, and it is noticeable that both PCA and Word2Vec still appear to be good alternatives for OOV items.

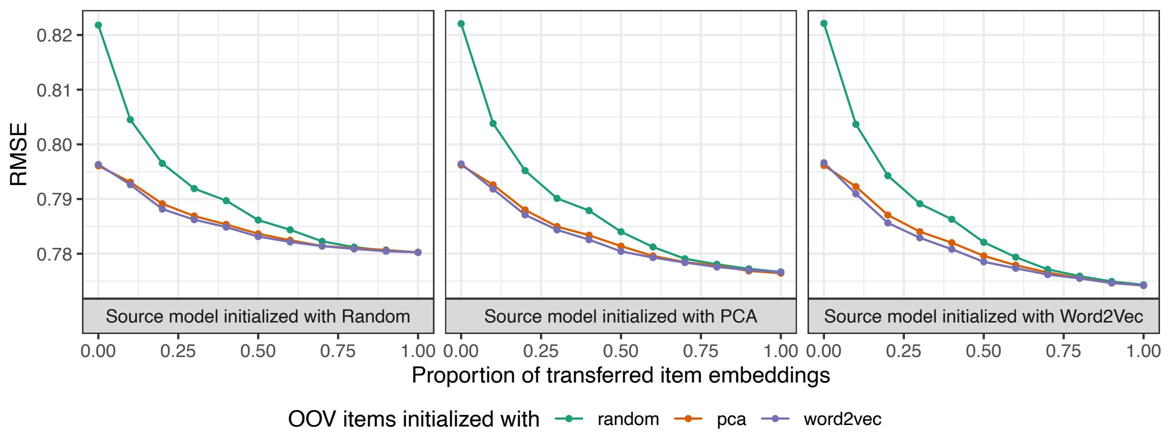

Figure 4.4: Predictive performance on MovieLens 100k using the *Source2Target* strategy considering MovieLens 24.9M as the source.

By summarising these results, we understand that the performance dramatically improves when even a few item representations are transferred from source to target, while the presence of OOV items prevents the model from achieving its best performance.

The same experiments were performed by extracting source and target samples from the other MovieLens datasets and are presented in Appendix B.

**RQ3. Does pre-training act as a model regularizer?**

To answer this question, we investigate whether pre-training leads to better local minima by employing the method proposed by Goodfellow et al. [13] for analyzing loss functions. The technique consists of a linear interpolation between two model states, which, in our case, are the randomly initialized MF and the various pre-trained MF models. Formally, the loss is evaluated by $\Theta = (1-\eta)\Theta_0 + \eta\Theta_1$, where $\Theta_0$ and $\Theta_1$ are the learned parameters of the baseline and the pre-trained model, respectively, and $\eta$ is the weight given to the pre-trained model.

We generated a set of 50 evenly spaced values over the $[0, 1]$ interval to represent varying values of $\eta$. The outcomes for the *Target2Target* scenario are illustrated in Fig. 4.5. It is noticeable that, in all cases, there is an elevation in the cost barrier between both solutions in training (blue curve) and testing (green curve). This observation implies that the pre-trained methodologies converge to distinct local minima compared to the baseline, except for MovieLens 100k pre-trained with Word2Vec, which showed signs of overfitting.

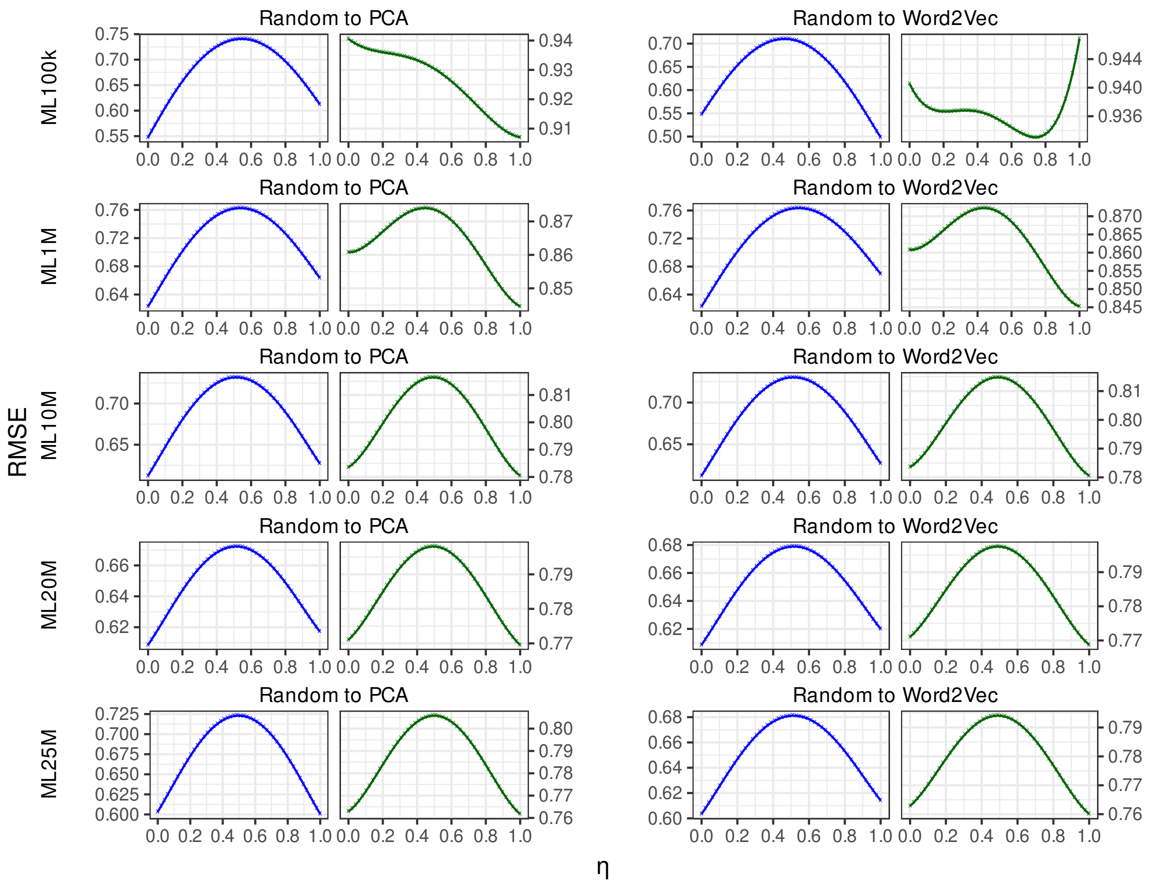

Figure 4.5: Loss landscape analysis of the Target2Target pre-training strategies. The $x$-axis contains the values of $\eta$, and the $y$-axis the RMSE. The concave shape of the curves indicates that models initialized randomly and initialized with pre-training reach different local minima. The observation is that pre-trained models have a higher training error than the randomly initialized ones, but a lower test error indicates the regularization effect.

Despite the pre-training strategies yielding worse train errors, particularly evident in the smaller datasets, they achieve better test errors. This observation suggests that pre-training not only facilitates convergence to more favorable local minima but also acts as a regularization mechanism, enhancing the generalization capability of the underlying model.

Fig. 4.6 illustrates the outcomes for the *Source2Target* scenario, focusing on the MovieLens 24.9M/100k source/target pair. We evaluate the scenarios where the source model is initialized with random, PCA, or Word2Vec embeddings, with subsequent transfer of the resulting representations to the target model for fine-tuning.

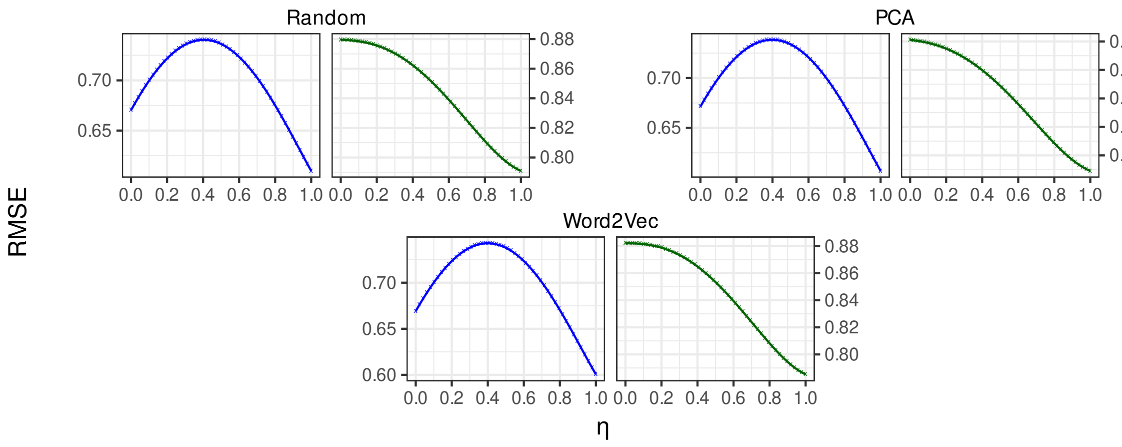

Figure 4.6: Loss landscape analysis for each initialization of MovieLens 24.9M in the *Source2Target* strategy. The plots show the cost barrier in the training curve and that achieving a smaller training error leads to a much smaller testing error, different from Target2Target, where pre-training leads to a higher training error with a smaller testing error.

In summarizing the findings for this question, it is evident that *RQ3* can be answered positively; pre-training leads MF toward more favorable local minima. Nevertheless, caution is advised to identify and prevent overfitting.

**RQ4. How do the best results achieved with pre-training compare to the best-known results on the corresponding datasets?**

To answer this last question, we used the Netflix Prize Dataset as the source and the MovieLens 100k and 1M as target datasets. First, we identified similar items across datasets using the procedure described in Sec. 3.3.2. After this process, we have two source Netflix datasets, one for each target dataset due to the different number of matched items. The statistics of these datasets were also reported in Table 4.1.

Fig. 4.7 recalls the same experiment performed in the MovieLens datasets, where we increase the proportion of transferred embeddings until reaching 100%, now having the original MovieLens 100k dataset as the target. We notice an analogous behavior to the previous one, but Word2Vec does not perform as well as expected.

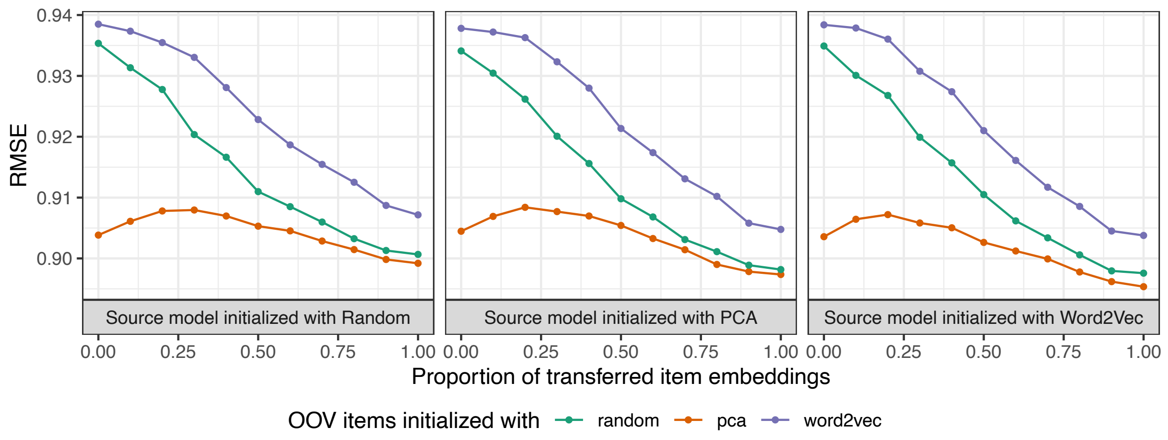

Figure 4.7: Predictive performance on MovieLens 100k using the *Source2Target strategy*, the source model being trained with $\approx$ 22M ratings from the Netflix Prize dataset.

To contextualize our findings with the state-of-the-art rating prediction, we now compare our best model employing the Netflix dataset as the source for MovieLens 100k[2] on the native u1-u5 splits as outlined in the leaderboard available on *Papers with Code* [3]. Our best setup was a source model pre-trained with Word2Vec, scoring in the top-4 position, around $1\%$ behind the best-performing method. Table 4.2 shows a version of the leaderboard that includes our best-performing setting and the Biased MF baseline.

For MovieLens 1M, this comparison is less direct since the leaderboard does not specify the size of the training and testing splits. However, considering the $80/20\%$ splits used in our experiment, our best model yielded an RMSE of $0.846$ against $0.868$ of the baseline.

Recalling *RQ4*, these observations confirm that pre-training alone may provide a good boost in placing a model as simple as vanilla MF without using side information on par with state-of-the-art models for this task.

[2] https://paperswithcode.com/sota/collaborative-filtering-on-movielens-100k
[3] https://paperswithcode.com/

Table 4.2: Papers with Code recommendation benchmarks for MovieLens 100k, adding our experiment best results.

| **Rank** | **Model** | **Source** | | **Target** | **RMSE** |
|---|---|---|---|---|---|
| | | **Initialization** | **Dataset** | | |
| 1 | GHRS [67] | - | - | - | 0.887 |
| 2 | GLocal-K [15] | - | - | - | 0.888 |
| 3 | MG-GAT [32] | - | - | - | 0.890 |
| 4(+) | **Source2Target MF** | **Word2Vec** | **Netflix** | **PCA** | **0.895** |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 14(+) | Baseline | - | - | Random | 0.935 |

## 4.2 Implicit Feedback

This section presents our findings to the same research questions, now considering Implicit Feedback and its models. In this case, we use the same number of embedding dimensions $d = 128$ and the number of epochs $e = 100$ while using the learning rate $\lambda = 0.0001$ for BPR and WBPR, as reported by Zhao et al. [69], and $\lambda = 0.003$ and $\alpha_0 = 0.01$ for WMF, as reported by Rendle et al. [46] as one of the best configurations for these models. While these works also search for the best number of dimensions and epochs, we used the same as in previous experiments to allow cross-task transferring.

We used explicit feedback datasets to perform the following experiments, converting them to binary matrices using the $t \geq 4$ threshold [29; 38], whereas recent studies use $t \geq 1$, predicting consumption, instead of likeness [20; 46]. While this makes sense, considering that implicit feedback datasets only capture the user interactions (positive), we chose to predict likeness since it better captures innate rating characteristics.

This section only reports the performance in terms of NDCG@10. However, we also report the values of other performance metrics in Appendix C.

There are a few differences between Rating Prediction and Item prediction when preparing the data to perform these experiments:

1. The most significant difference is that cross-validation is not standard in this case since

researchers often favor one of the following strategies: a hold-out, a split considering a period, or a split considering the number of interactions [58].

2. A second difference is that we cannot use PCA as a *Target2Target* initialization strategy since it performs poorly in binary matrices. Therefore, we reduce our problem by comparing random, self-supervised, and supervised pre-training.

3. Another difference is that most methods behave differently, considering how the rating is transformed into a binary value, as pointed out in subsection 2.2.3.

4. Lastly, the evaluation metrics are different: we only had RMSE for Rating prediction, and now we use a set of metrics to measure how good the predictions are and how well the model ranks them: Precision@$k$, Recall@$k$, and NDCG@$k$.

Concerning the first question, we opted for using hold-out since it is the most frequent and easy data-splitting method, assigning the same $80\%/20\%$ train/test proportions as in rating prediction. We also explore the third difference in assessing the *Target2Target* scenario.

To avoid having issues and poor implementations of the models and metrics, we took advantage of the publicly available *Cornac* [50] library[4], which includes all the models and metrics of interest.

**RQ1. Does pre-training (task- or non-task-specific) improve a model's predictive performance?**

We answer this question in two ways: the first is inspired by one of the suggestions made by Verachtert et al. [58], where only the most recent $N$ interactions of a dataset are used. Since MovieLens 25M is the most recent dataset we have been working with, we'll use it as the data source for this experiment. To be concise with the rating prediction experiments, we define $\{100\text{k}, 1\text{M}\}$ as the sample sizes.

Figure 4.8 shows the results of this experiment in terms of NDCG@10. Although WBPR does not benefit from pre-training, we see performance improvement for both BPR and WMF. However, for NLP, it is known that the quality of latent representations generated by Word2Vec increases along the input size, which, in this case, is very limited.

[4]Cornac library GitHub repository

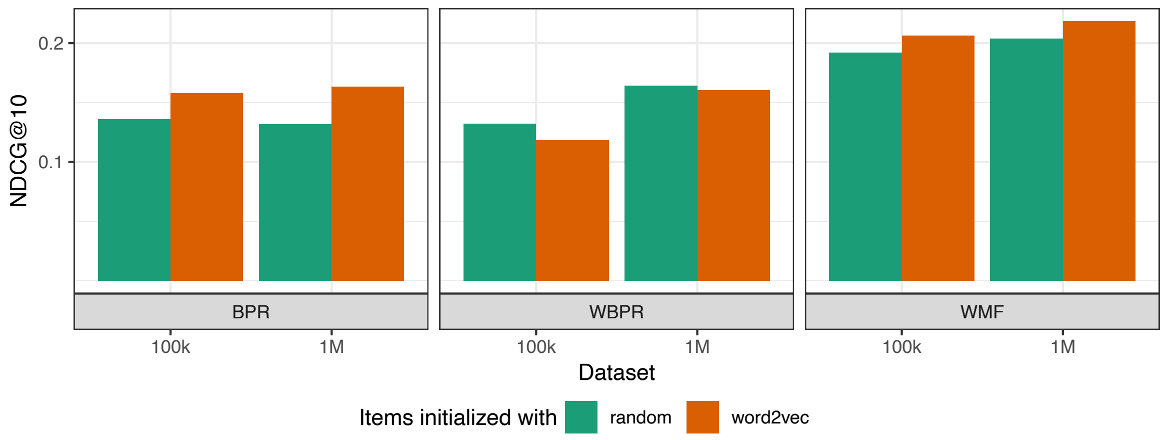

Figure 4.8: Predictive performance on samples extracted from the MovieLens 25M dataset using the *Target2Target strategy*

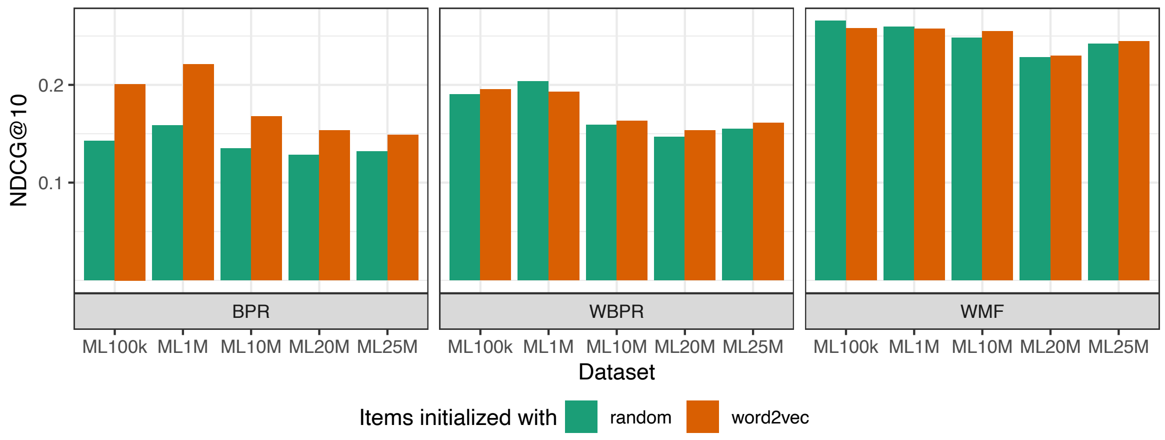

Figure 4.9: Predictive performance on the MovieLens datasets using the *Target2Target strategy*

The second method is the one performed for rating prediction, evaluating the performance of the models in all the MovieLens datasets. Figure 4.9 shows the results for the original MovieLens datasets. Pre-training with Word2Vec improves BPR performance significantly, while WBPR and WMF only benefit from it on larger datasets, which is expected, considering that increasing data can result in a regularization effect [13]. As in rating prediction, the impact of *Target2Target* pre-training also diminishes as the dataset size increases, which is noticeable, especially for BPR.

**RQ2. How do out-of-vocabulary items impact a pre-trained model's predictive performance?**

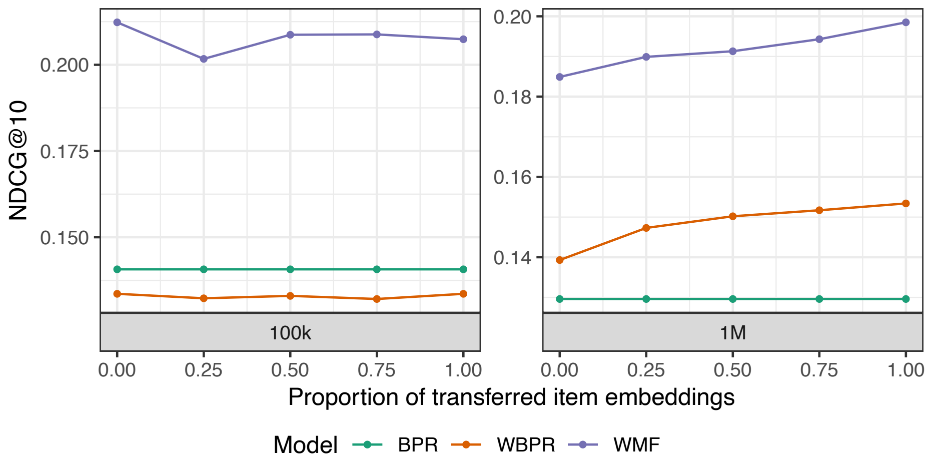

Figure 4.10: Predictive performance on the 100k and 1M samples of the MovieLens 25M dataset using the *Source2Target strategy*

For the *Source2Target* scenario, supported by our findings in the rating prediction task, we chose not to iterate through all the MovieLens datasets. Instead, we only evaluated the transfer performance between the MovieLens 24.9M/100k and 24M/1$M$ source/target pairs. We also simplified this procedure by randomly initializing the OOV item representations since the complete transfer in rating prediction showed no statistical difference between the fallback initialization. Figure 4.10 depicts our findings, already exploring restricting the transferred items. The lack of improvement for the 100k sample is unusual and conflicts with our expectations. However, for the 1M sample, WMF and WBPR improve as more item embeddings are transferred, while BPR remains almost unchanged in all cases.

However, we do not have enough evidence to conclude that supervised pre-training guarantees performance improvement since not all cases had a positive outcome. Recalling *RQ2.*, we can affirm that OOV items prevent the models from achieving their best performance.

### RQ3. Does pre-training act as a model regularizer?

The methodology for executing our experiments has changed between rating prediction and item prediction, as well as the packages we use. The most significant differences are (1) the library used to train and score the models and (2) the train/test loss evaluation. Due to these differences, we cannot retrieve and interpolate the losses of the two models using the method proposed by Goodfellow, mainly because evaluating the performance metrics during the training stage highly increases the execution time. A second reason is that the performance metric differs from the models' loss function.

The premise of pre-training is that the models should achieve a lower loss without overfitting, which leads to metric improvement. Figure 4.11 depicts the final loss of each model as the percentage of transferred embeddings increases. It is noticeable that BPR's loss remains unchanged while WBPR (bigger is better) and WMF (lower is better) improve. This finding corroborates the hypothesis that supervised pre-training also results in performance improvement.

Figure 4.12 helps us understand the difference between the losses of a randomly initialized model and a pre-trained one by plotting the loss of the pre-trained model over the randomly initialized model loss. We only notice a significant improvement in the WBPR loss for both samples. However, as aforementioned, the WMF performance improves while keeping a similar loss, which indicates better local minima.

Recalling *RQ3.*, we conclude that pre-training leads to better local minima since both WBPR and WMF have performance improvement while maintaining or improving the loss.

### RQ4. How do the best results achieved with pre-training compare to the best-known results on the corresponding datasets?

To answer *RQ4.*, we again recall the *Papers with Code* benchmark. However, since the MovieLens 100k dataset was released over 20 years ago, it is more commonly used for the rating prediction task. We have a weak comparison base for the other datasets since most

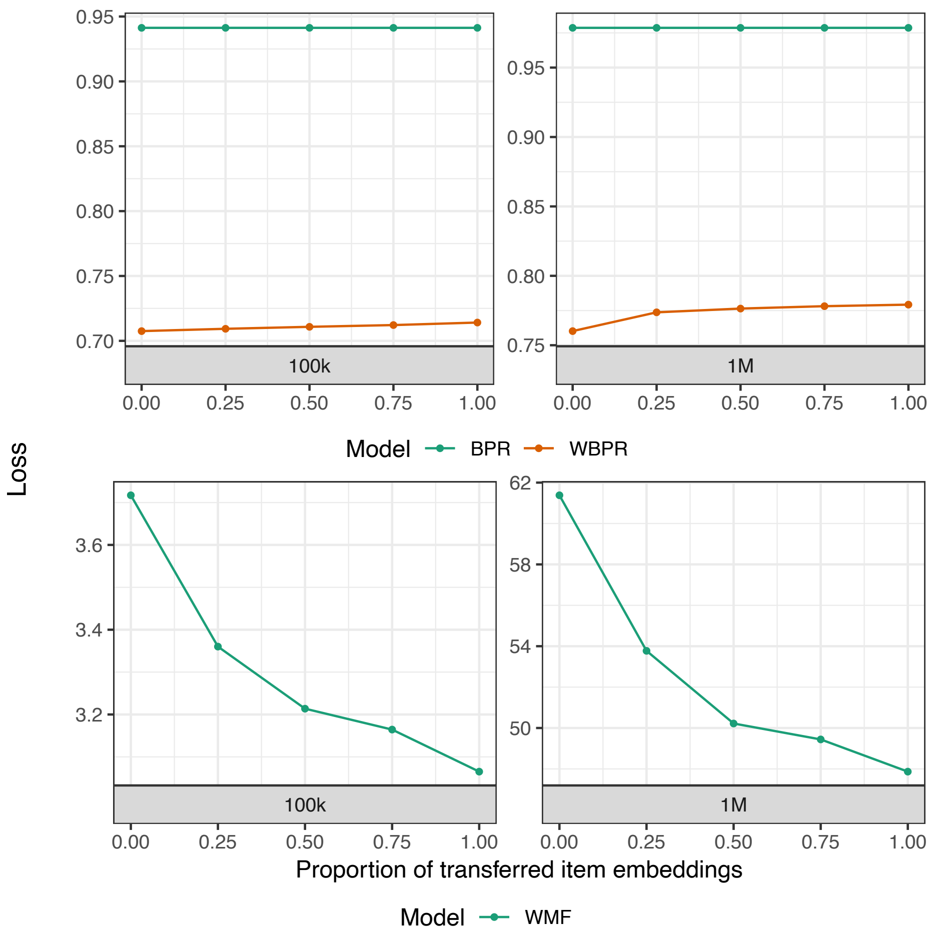

Figure 4.11: Final loss for each model on the 100k and 1M samples of the MovieLens 25M dataset using the *Source2Target strategy*. Since the WMF loss differs from the BPR and WBPR, we separated them into different rows.

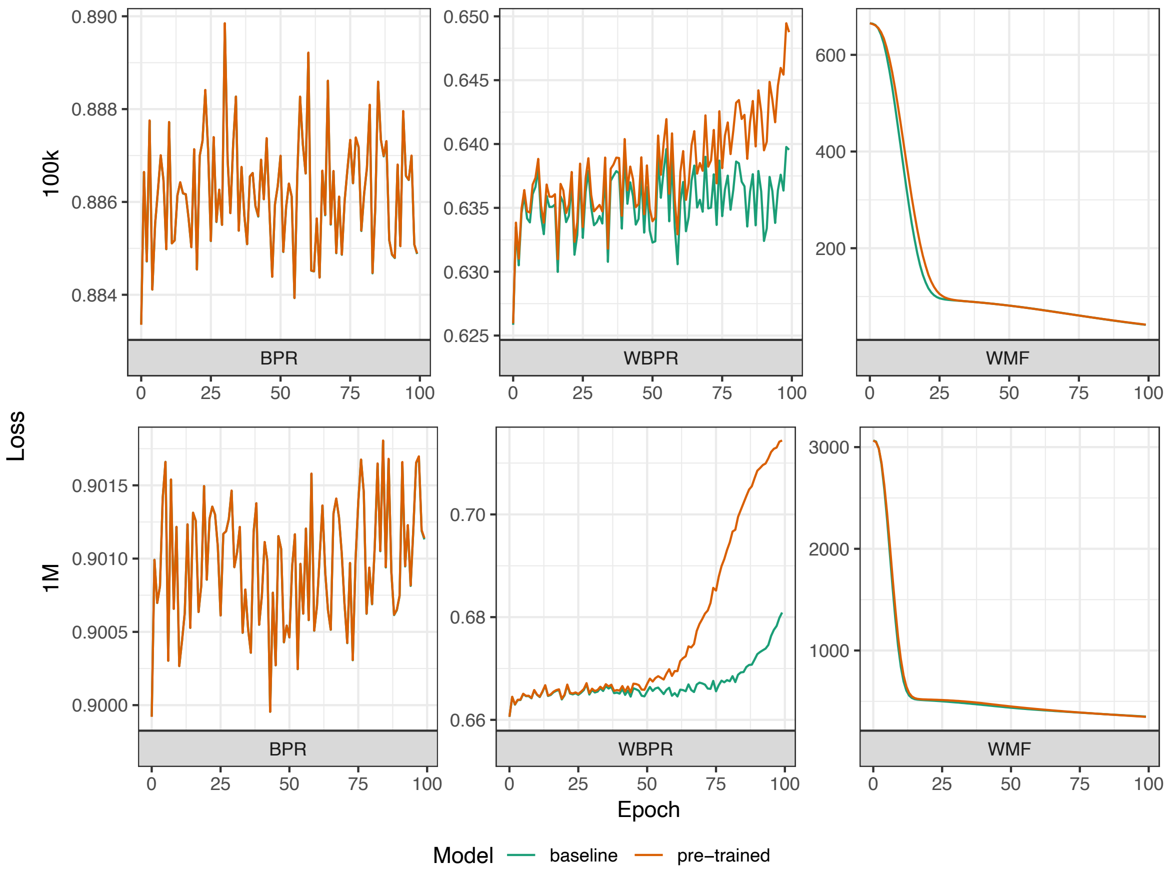

Figure 4.12: Pre-trained model loss over the baseline's loss on the 100k and 1M MovieLens 25M dataset samples. For BPR, both baseline and pre-trained losses completely overlap.

perform a hold-out, and the test size is also varied. Still, we executed the transfer again using the Netflix Prize dataset as the source to the MovieLens 100k and 1M datasets.

Figure 4.13 illustrates this experiment. We find that the performance of WBPR improves with pre-training, although the WMF performance doesn't, which is unexpected considering the previous *Source2Target* experiments. Again, BPR oddly remains unchanged.

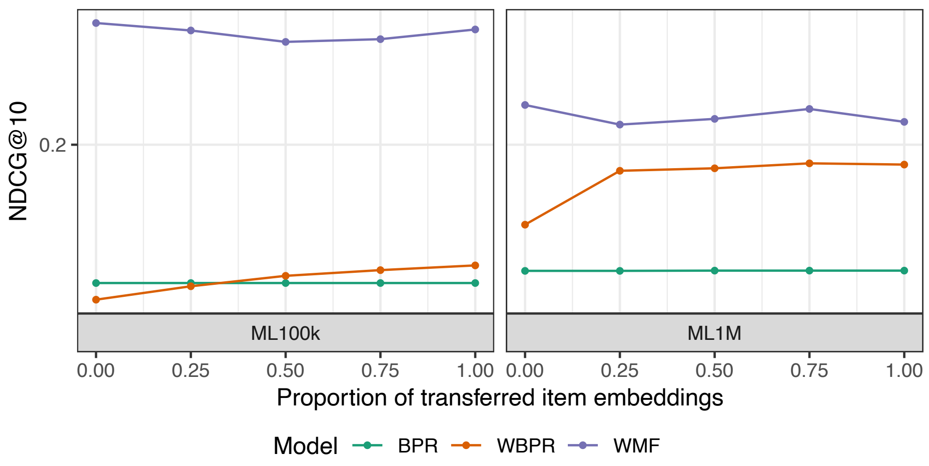

Figure 4.13: Predictive performance of models pre-trained with Netflix embeddings and evaluated in the MovieLens 100k and 1M datasets using the *Source2Target strategy*

Again, we compare the loss of the pre-trained model with the baseline loss to further understand the impact of pre-training. This comparison is illustrated in Figure 4.14. We can see that WBPR achieves a significantly lower loss while WMF almost does not present differences between them. It is fair to assume that the local minimum achieved by WMF is worse than that obtained by random initialization, but the overfitting hypothesis is also valid. The losses from the baseline and pre-trained versions of BPR overlap again.

Table 4.3 shows the *Papers with Code* leaderboard, including the baseline and our best-performing model. Unlike rating prediction, we do not achieve relevant results in this case, and the baseline also performs better than our best pre-trained model. We notice that all models in the leaderboard were proposed after 2016, and the top-performing methods use more complex strategies, such as Sequential Recommendation.

Since newer MovieLens datasets are mainly used for ranking prediction, we can also compare the performance achieved by our best-performing *Target2Target* model on more recent versions. Table 4.4 shows the *Papers with Code* leaderboard for MovieLens 20M since

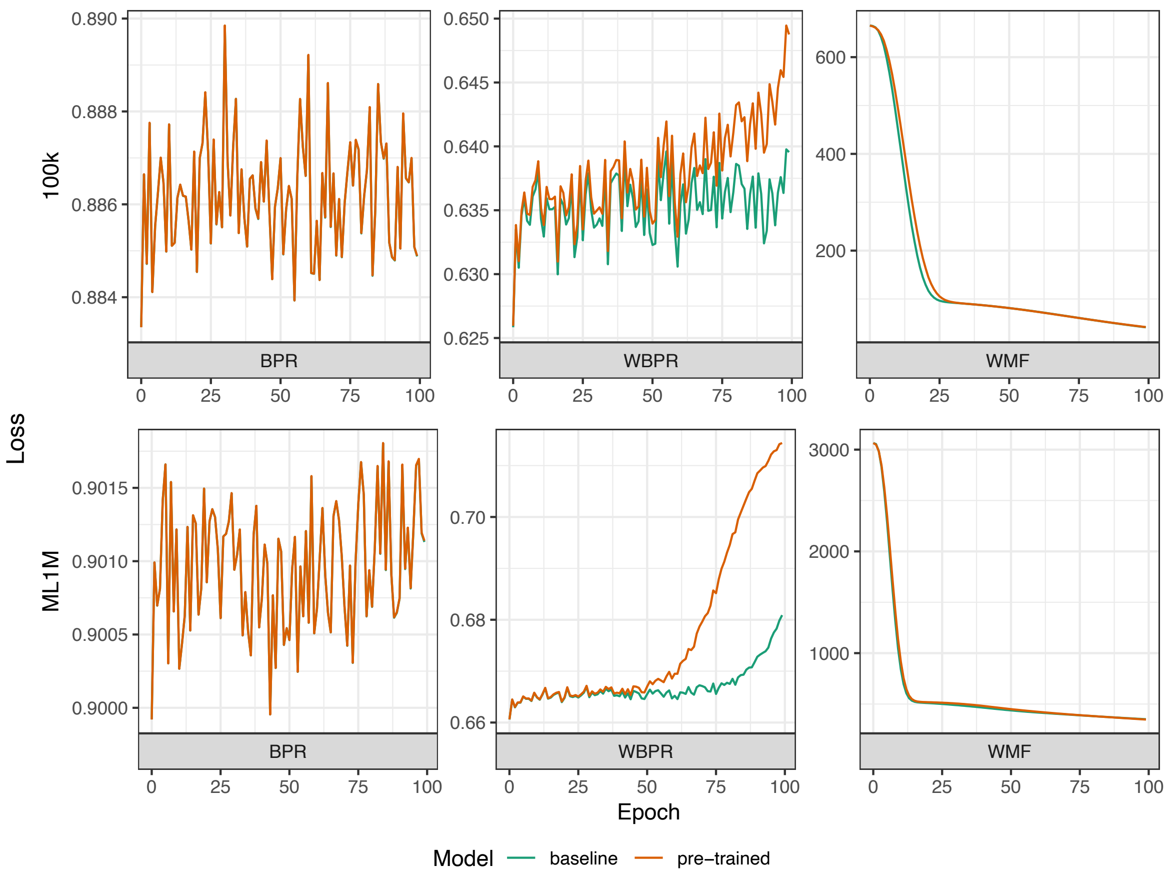

Figure 4.14: Netflix pre-trained model loss over the baseline's loss on the ML100k and ML$1M$ datasets. Again, both baseline and pre-trained losses completely overlap for BPR.

Table 4.3: Papers with Code recommendation benchmarks for MovieLens $1M$, adding the baseline and our best-performing model.

| **Rank** | **Model** | **NDCG@10** | **Year** |
|---|---|---|---|
| 1 | SSE-PT [64] | 0.6292 | 2019 |
| 2 | SASRec [28] | 0.5905 | 2018 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 8(+) | Baseline | 0.2654 | 2024 |
| 9(+) | **Target2Target WMF** | 0.2574 | 2024 |
| 10(+) | **Source2Target WMF** | 0.2144 | 2024 |

there still isn't a leaderboard for the newer MovieLens 25M. Unlike the previous leaderboard, more entries report Recall@50 instead of NDCG@10, so we compare the results of this metric. In this leaderboard, all the models were also proposed after 2016, and most of them use complex models, especially Autoencoders, and some others report using side information.

Table 4.4: Papers with Code recommendation benchmarks for MovieLens 20M, adding the baseline and our best-performing model.

| **Rank** | **Model** | **NDCG@10** | **Year** |
|---|---|---|---|
| 1 | RecVAE [53] | 0.553 | 2019 |
| 2 | VASP [57] | 0.552 | 2021 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 10(+) | Baseline | 0.4488 | 2024 |
| 11(+) | **Target2Target WMF** | 0.4476 | 2024 |

When looking at the entire pipeline and its results, we can raise a few hypotheses to understand why *Source2Target* hasn't performed as well as in rating prediction.

First, some models do not differ between missing and negative interactions and discard the negative ones. This is the case of BPR, and we can imagine the impact by measuring the number of interactions where $t \geq 4$ in the samples extracted from the MovieLens datasets: for the MovieLens 24.9M/100k source/target pair, $\approx 40\%$ of the interactions are removed, whereas the MovieLens 24M/1M source/target pair has $\approx 36\%$ of the data removed. This can also lead to removing the core property of the dataset kept when sampling the source/target pairs from the original MovieLens 25M, leading to weaker user/item representations.

Second, we are converting an explicit feedback dataset into an implicit feedback one. While this binarization process captures the overall intention of rating, i.e., the user liked or not that item, it may also lose information. To validate this hypothesis, we again performed the *Source2Target* experiment for item consumption with the $t \geq 1$ threshold instead of likeness prediction. As shown in Figure 4.15, WBPR presents the same behavior as BPR for likeness prediction since distinguishing negative items and missing interactions is their most significant difference, and, in this case, it doesn't exist. Unlike the previous scenarios, WMF has deteriorated performance instead of improving, especially for the 100k sample. One possible cause for this outcome is that binarizing for item consumption degrades WMF's loss function, given its similarity to the one employed by MF for rating prediction.

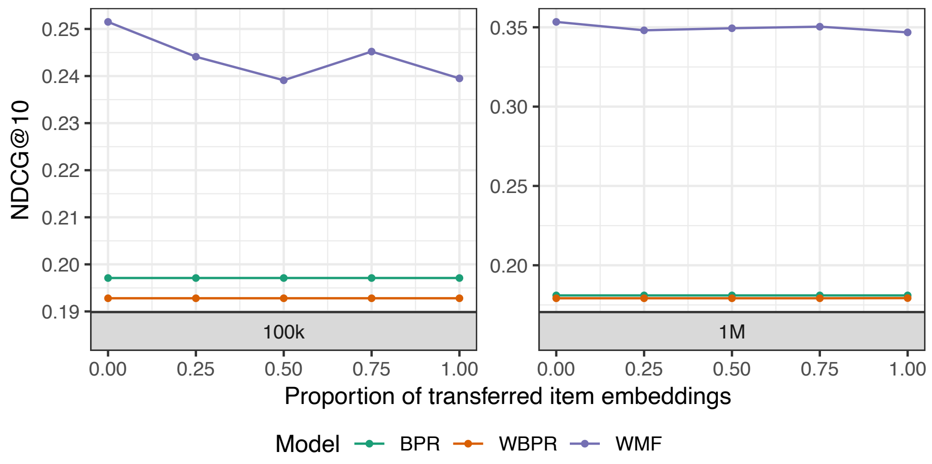

Figure 4.15: Predictive performance of models pre-trained and evaluated using the MovieLens 24.9M/100k and 24M/1M source/target pairs with the *Source2Target* strategy for item consumption

## 4.3 Cross-Task Transfer Learning

In this section, we report our brief findings for cross-task transfer learning, where we either train the source models for rating prediction and transfer the learned representations for ranking prediction models or train for ranking prediction and transfer to rating prediction models. It is worth pointing out that this is only possible for the *Source2Target* scenario since it is supervised pre-training. We include supplementary figures reporting performance in other metrics in Appendix D

Figure 4.16 shows the results for the *Ranking2Rating* case. We notice that, except for BPR, all models lead to a significative performance improvement regarding RMSE. This is especially interesting since WBPR uses different optimization criteria, while WMF uses optimization criteria similar to MF (i.e., quadratic loss).

Figure 4.17 illustrates the results for *Rating2Ranking*. Noticeably, for both sample datasets, the WBPR slightly increases its performance. WMF, however, only benefits from

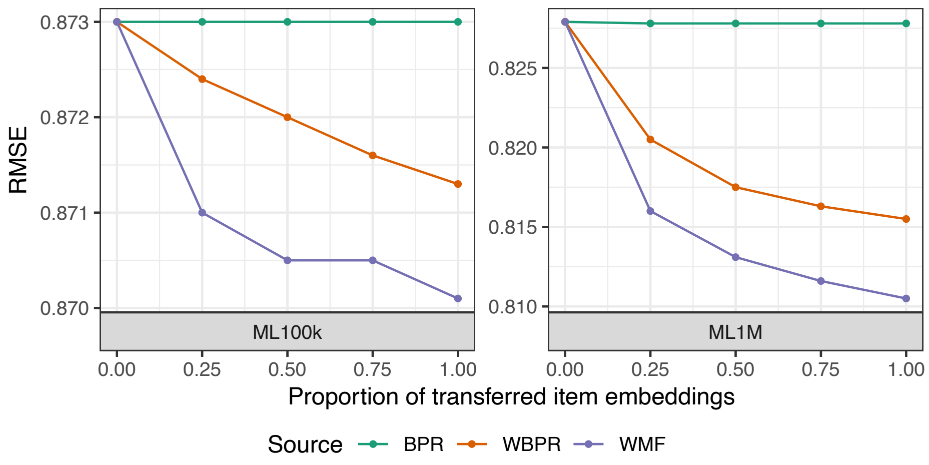

Figure 4.16: Predictive performance on the 100k and 1M samples of the MovieLens 25M dataset using the *Ranking2Rating strategy*

it for the 1M sample, while the slight decrease in performance for the 100k sample may indicate overfitting.

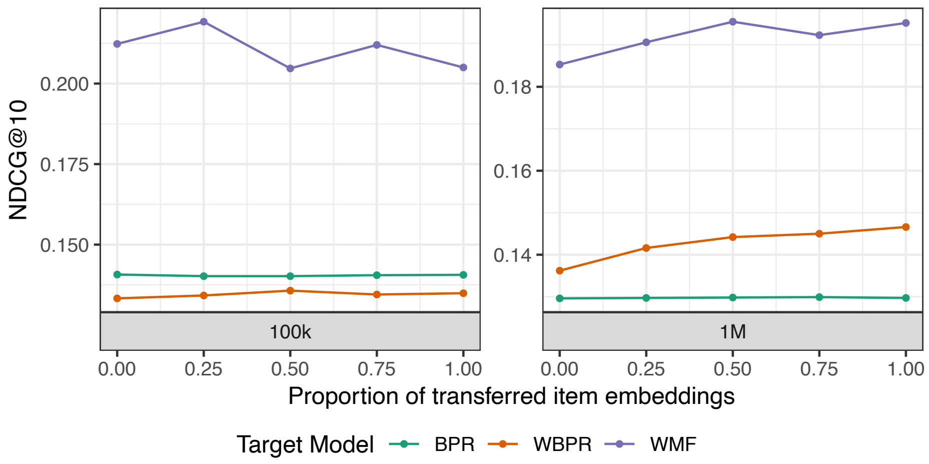

Figure 4.17: Predictive performance on the 100k and 1M samples of the MovieLens 25M dataset using the *Rating2Ranking strategy*

Figure 4.18 shows the pre-trained model loss against the baseline's loss to verify this hypothesis. We can see that, for both datasets, WMF has a slightly lower loss, while WBPR significantly benefits from pre-training. However, since WMF didn't improve performance for the 100k sample, it is fair to say that the model achieved a worse local minima compared to the baseline, but the overfitting hypothesis is also valid.

Comparing the results of both cross-task scenarios, *Ranking2Rating* led to performance

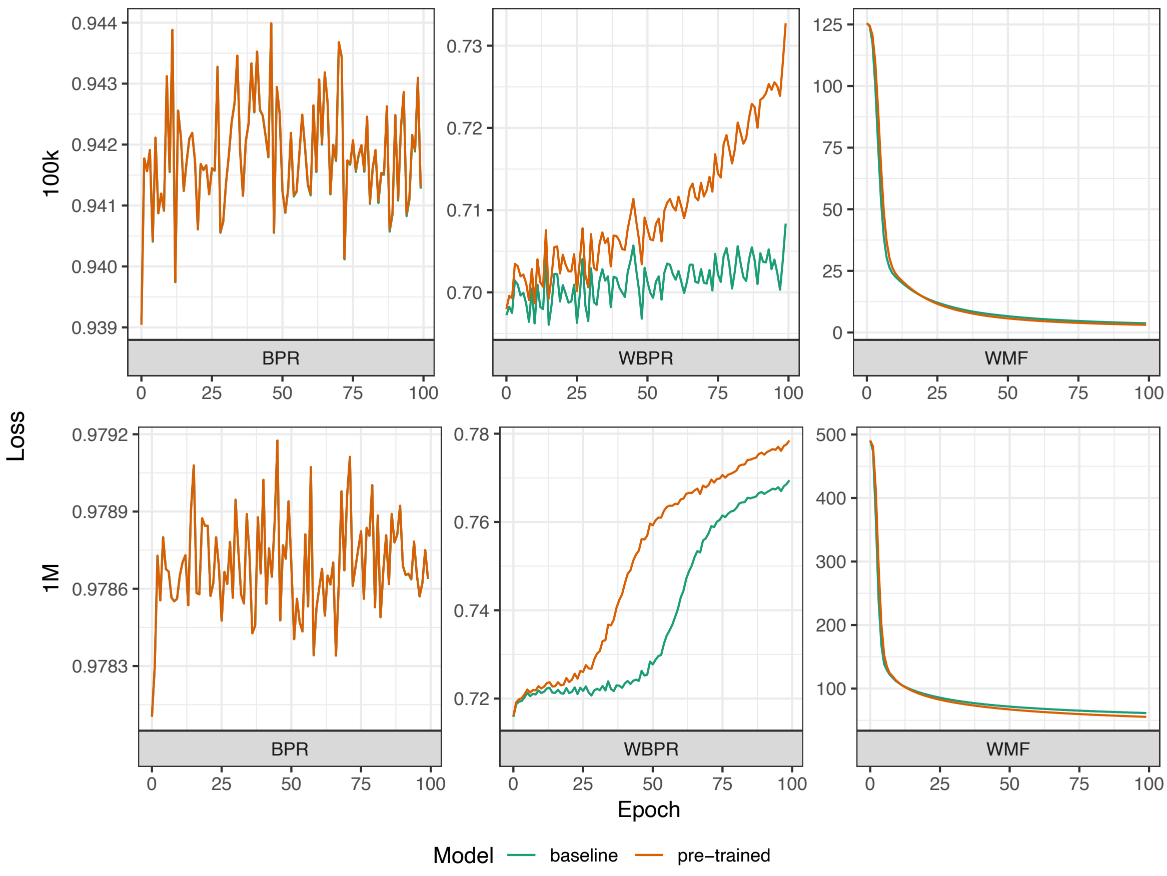

Figure 4.18: Pre-trained model loss over the baseline's loss on the 100k and $1M$ MovieLens 25M dataset samples. For BPR, both baseline and pre-trained losses completely overlap.

improvement for both datasets, except for BPR. However, for *Rating2Ranking*, the impact of pre-training is much smaller. We hypothesize that this happens due to the binarization process with the $t \geq 4$ threshold, which captures the essence of the dataset beforehand and does not benefit from the rating signal. Since binarization captures the most innate rating characteristics, the same hypothesis makes sense for *Ranking2Rating*. At the same time, training in the target dataset fine-tunes the latent representations for the rating prediction task.

# Chapter 5

# Conclusions and Future Work

In light of the continuous increase of pre-training and transfer learning usage for machine learning, we proposed the *Target2Target* and *Source2Target* pre-training strategies to explore these techniques for RSs in a wide variety of MovieLens datasets and the Netflix Prize dataset. We also performed ablation studies to understand the impact of OOV items and applied these techniques to real transfer learning scenarios. To the best of our knowledge, this work and the

The following sections summarise our contributions and findings, present our conclusions, and discuss ideas for future research.

## 5.1 Summary of Contributions

In this section, we summarize the main contributions of this work.

1. In Chapter 3, we define a set of possible pre-training methods, considering unsupervised, self-supervised, and supervised strategies. Evaluating these different techniques allows us to explore which may lead to better results, depending on the problem and available data.

2. Also in Chapter 3, we present an experimental setup to solve the problem of matching items across datasets, allowing pre-training and transfer learning to be executed. In doing so, we enable researchers to perform similar techniques and achieve superior performance when facing a lack of problem-specific data.

3. Chapter 4 presents our findings for the *Target2Target* scenario, which leads to performance improvement in all cases for rating prediction except Word2Vec for MovieLens 100k and also increases performance for ranking prediction on larger datasets.

4. In Chapter 4, we also evaluate the *Source2Target* scenario, considering both same- and cross-task transfers, which leads to performance improvement in all cases of rating prediction but doesn't guarantee performance improvement for ranking prediction. The cross-task transfers are compelling since they allow explicit feedback datasets to improve performance on an implicit feedback task and vice-versa.

5. Lastly, we thoroughly evaluate the impact of OOV items for rating and ranking prediction in the *Source2Target* scenario. In doing so, we can better understand when to prioritize unsupervised and self-supervised pre-training over the supervised strategy.

## 5.2 Conclusions

In this work, we presented three and two pre-training strategies for rating and ranking prediction, respectively, summarized in the *Target2Target* and *Source2Target* scenarios. Our approaches are based on the premise that unsupervised and self-supervised maintain innate data characteristics, such as similarity. At the same time, the supervised approach takes advantage of the learned characteristics of the data that are also present in other datasets. While the use of these techniques was proposed in the context of this work, we encourage further experimentation with other existing unsupervised and self-supervised techniques and verify if the supervised technique leads to improvement with more complex methods, such as Autoencoders and DNN-based models. Since we prioritized using the best hyperparameter values reported in the literature, we also encourage searching for the best setting for each problem and model.

## 5.3 Future Work

We plan to investigate the impact of pre-training in other domains, such as music, where we find a wide variety of datasets publicly available, with a special interest in native implicit feedback datasets, avoiding eventual side-effects of binarization. We also aim to research and verify if the supervised technique improves other types of RSs, especially Sequential- and Graph-based recommenders. This is particularly interesting considering information-restricted environments, such as the Session-based recommendation scenario, where the user cannot be identified. Lastly, part of our initial findings, published in IUI'2022, showed that pre-training MF models with PCA creates a trade-off between model performance and explainability. Considering the other pre-training strategies used in this research, this is also an interesting path to explore further.

# Appendix A

# Algorithms

**Algorithm 1:** MovieLens Data splitting algorithm

```
Input: $D$: MovieLens 25M ratings; $U$: Unique user IDs; $t$: Target Size;
Result: $S$: Source ratings; $T$: Target ratings
$S, T, T_u \leftarrow [], [], \{\}$; $R_u \leftarrow randomize(U)$; $c = 0$;
for $u \in R_u$ do
    if $c \geq t$ then break;
    $T_u \leftarrow T_u \cup u$;
    for $uid, iid, r, ts \in D$ do
        if $uid = u$ then $T \leftarrow T \cup (uid, iid, r, ts)$; $c = c + 1$;
    end
end
$S_u = U - T_u$;
for $u \in S_u$ do
    for $uid, iid, r, ts \in D$ do
        if $u = uid$ then $S \leftarrow S \cup (uid, iid, r, ts)$;
    end
end
```

# Appendix B

# Explicit Feedback Supplementary Material

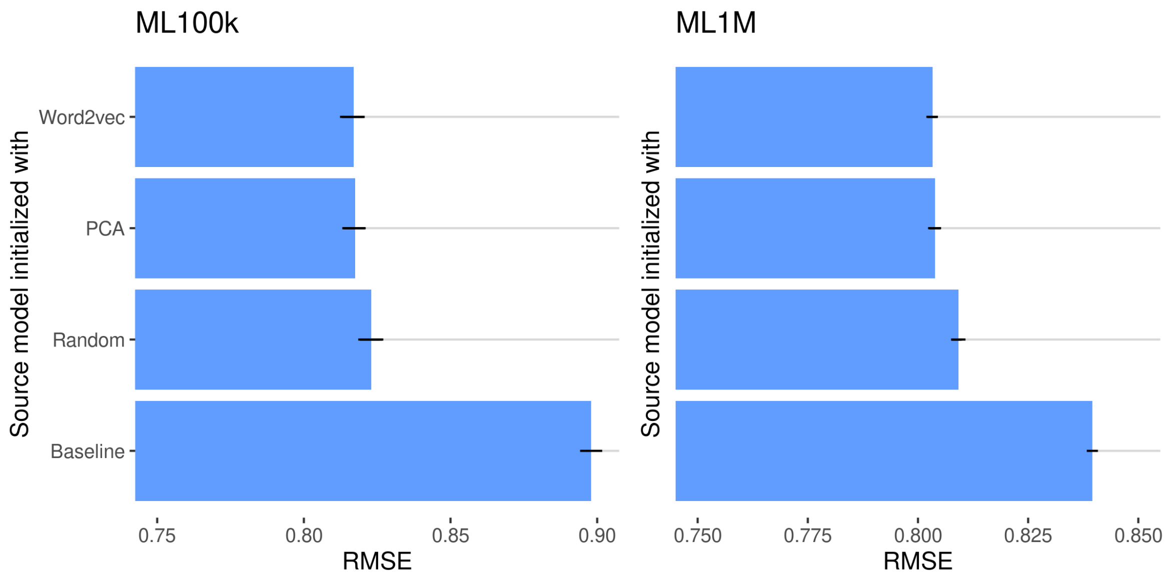

Figure B.1: Predictive performance on samples of MovieLens 10M in the *Source2Target* pre-training scenario. Error bars denote the standard deviation from 5-fold cross-validation.

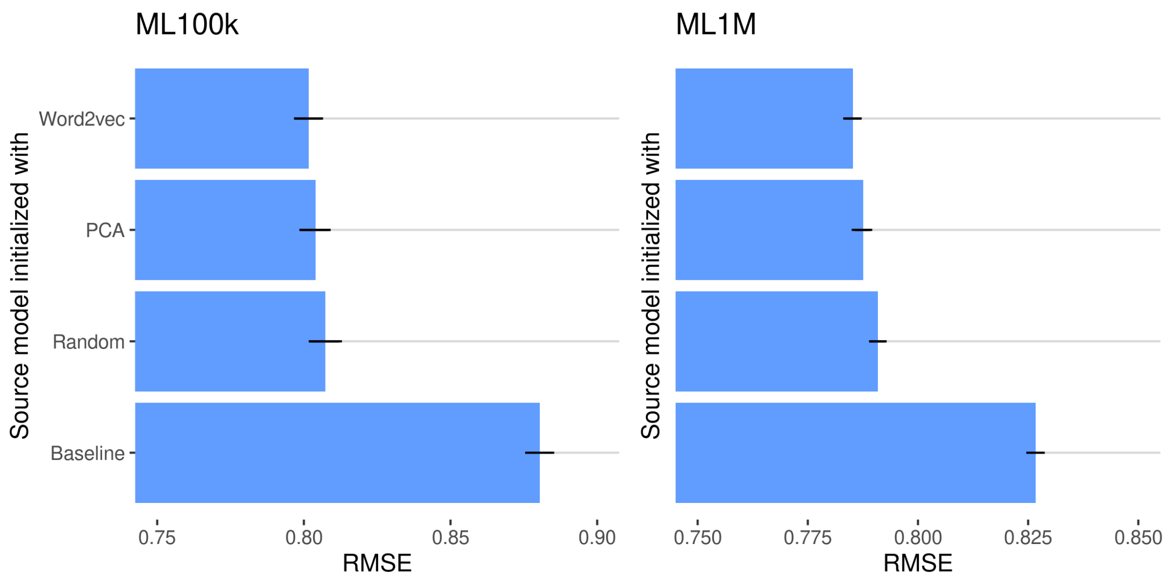

Figure B.2: Predictive performance on samples of MovieLens $20$M in the *Source2Target* pre-training scenario. Error bars denote the standard deviation from $5$-fold cross-validation.

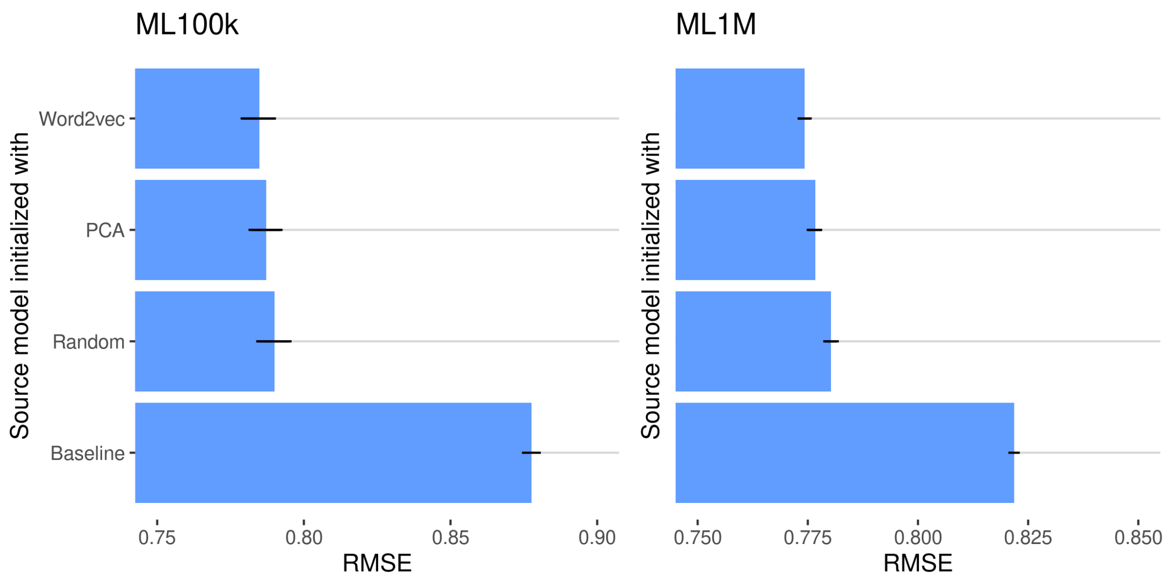

Figure B.3: Predictive performance on samples of MovieLens $25$M in the *Source2Target* pre-training scenario. Error bars denote the standard deviation from $5$-fold cross-validation.

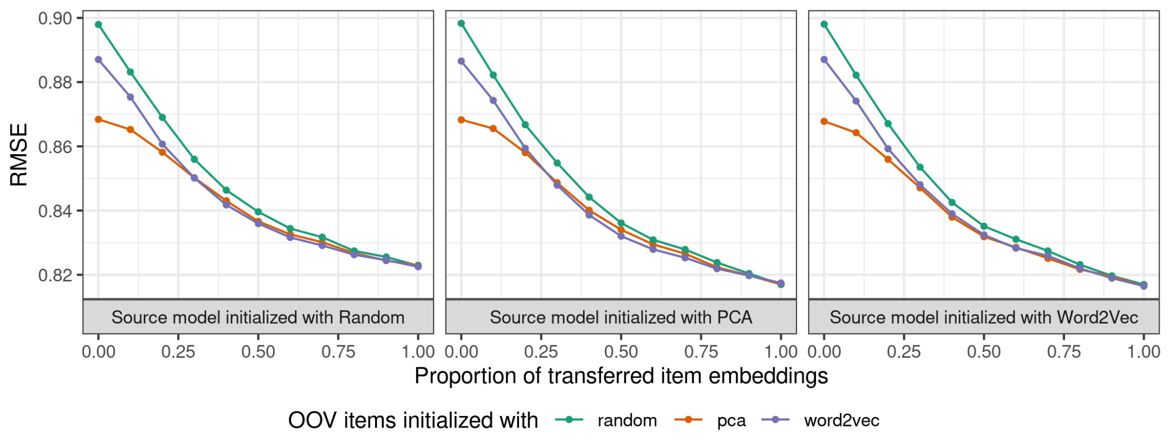

Figure B.4: Predictive performance on MovieLens 100k using the *Source2Target* strategy considering MovieLens 9.9M as the source.

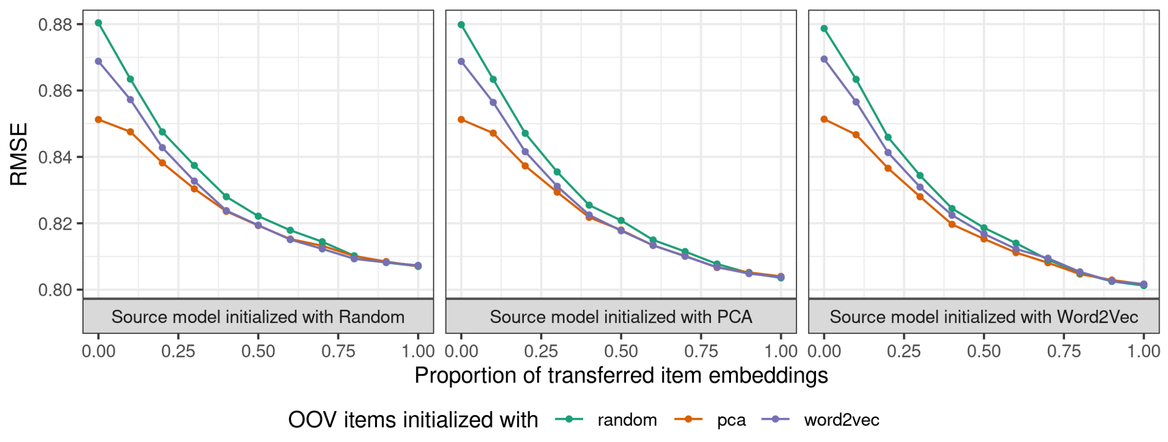

Figure B.5: Predictive performance on MovieLens 100k using the *Source2Target* strategy considering MovieLens 19.9M as the source.

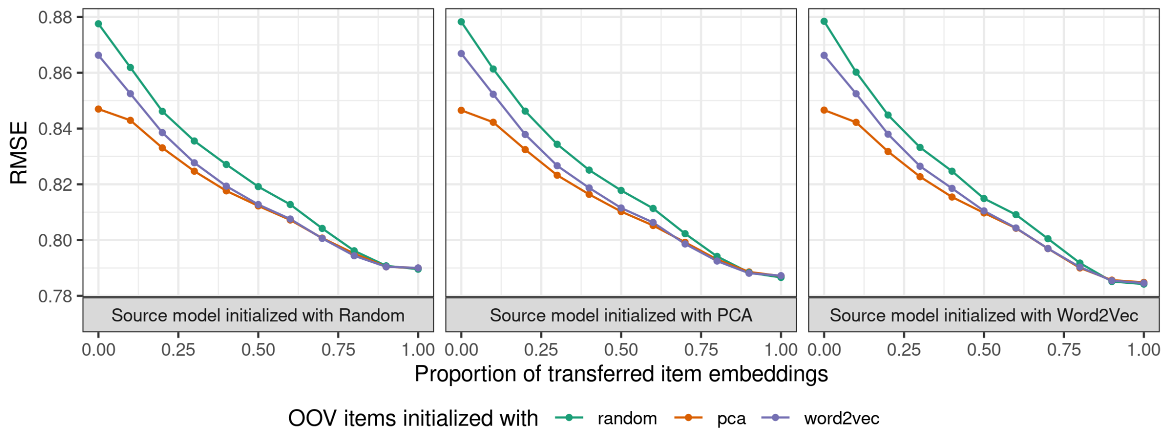

Figure B.6: Predictive performance on MovieLens 100k using the *Source2Target* strategy considering MovieLens 24.9M as the source.

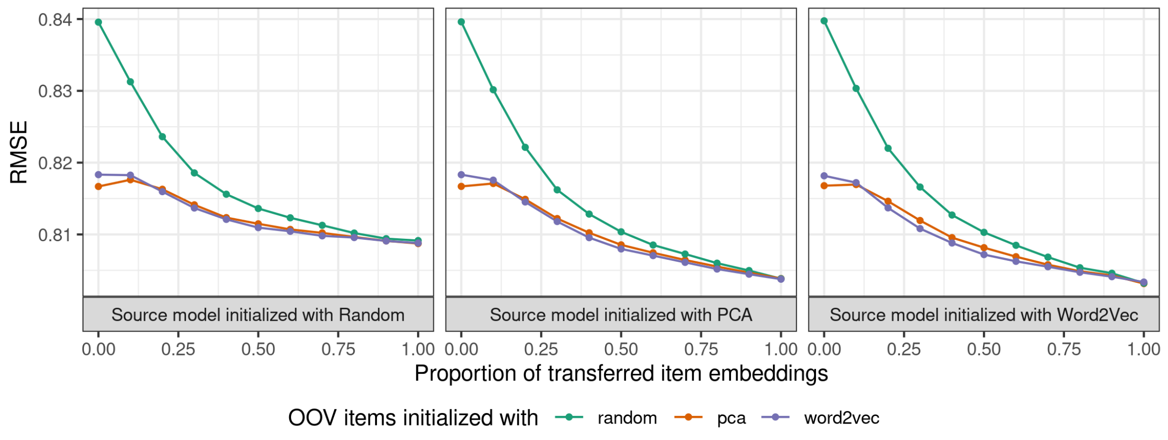

Figure B.7: Predictive performance on MovieLens 1M using the *Source2Target* strategy considering MovieLens 9M as the source.

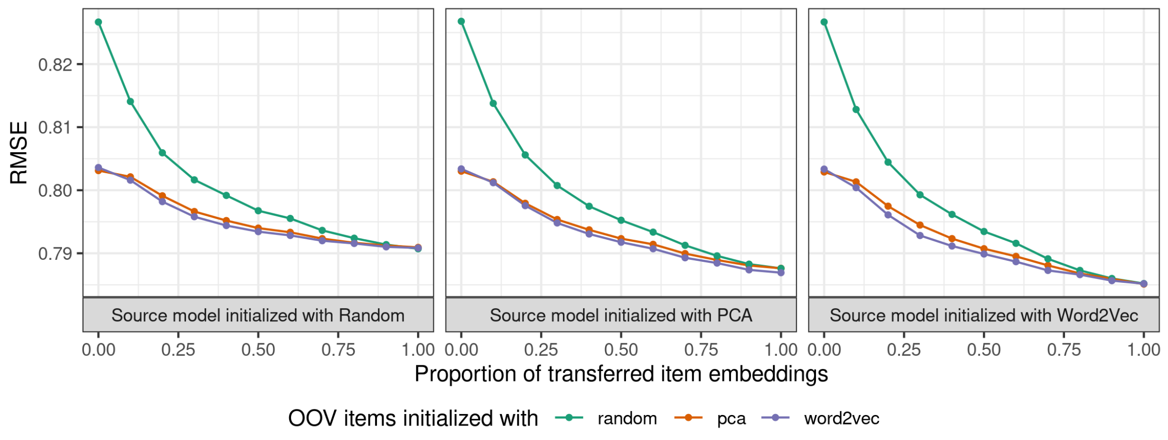

Figure B.8: Predictive performance on MovieLens 1M using the *Source2Target* strategy considering MovieLens 19M as the source.

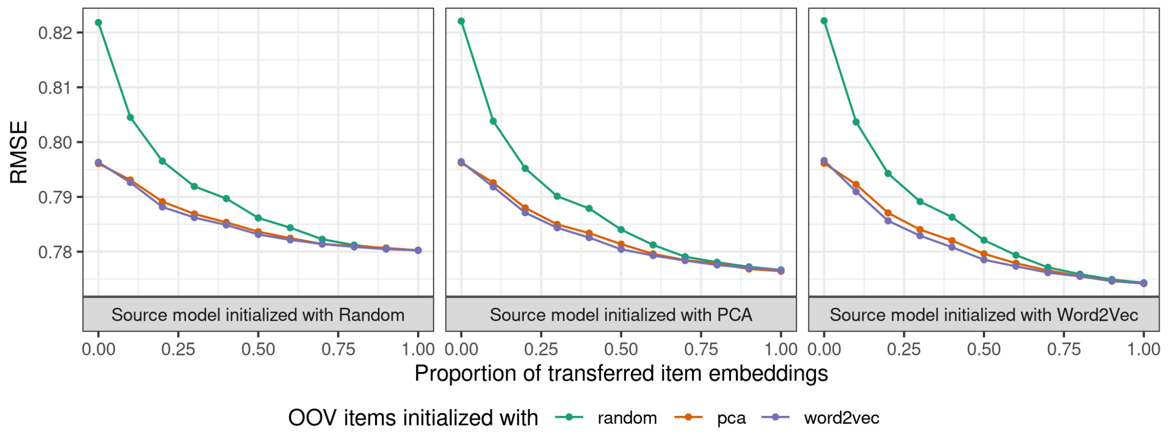

Figure B.9: Predictive performance on MovieLens 1M using the *Source2Target* strategy considering MovieLens 24M as the source.

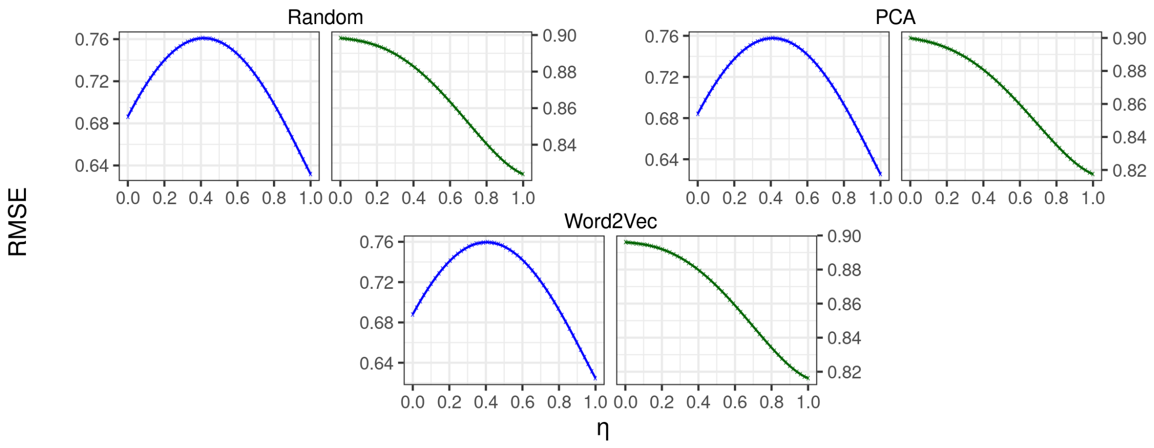

Figure B.10: Loss landscape analysis on a 100k sample for each initialization from MovieLens 9.9M in the *Source2Target* strategy.

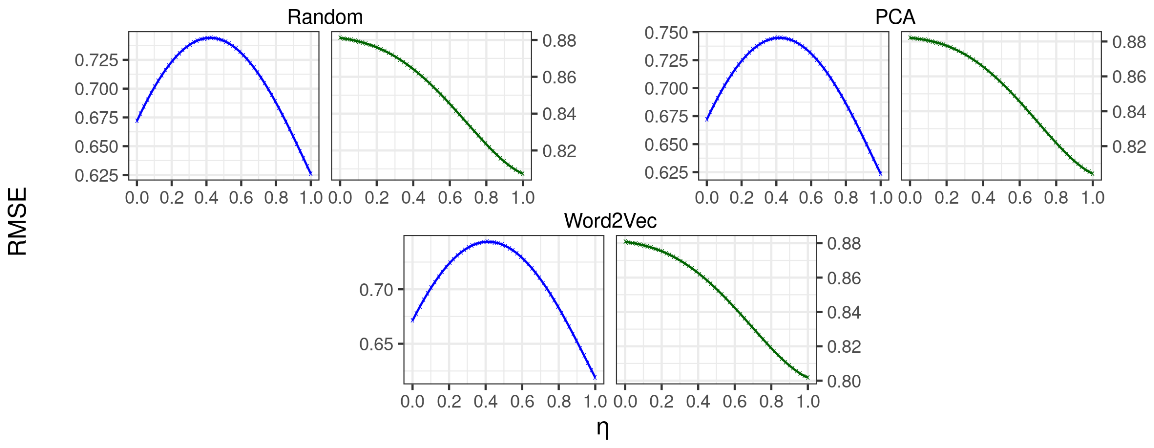

Figure B.11: Loss landscape analysis on a 100k sample for each initialization from MovieLens 19.9M in the *Source2Target* strategy.

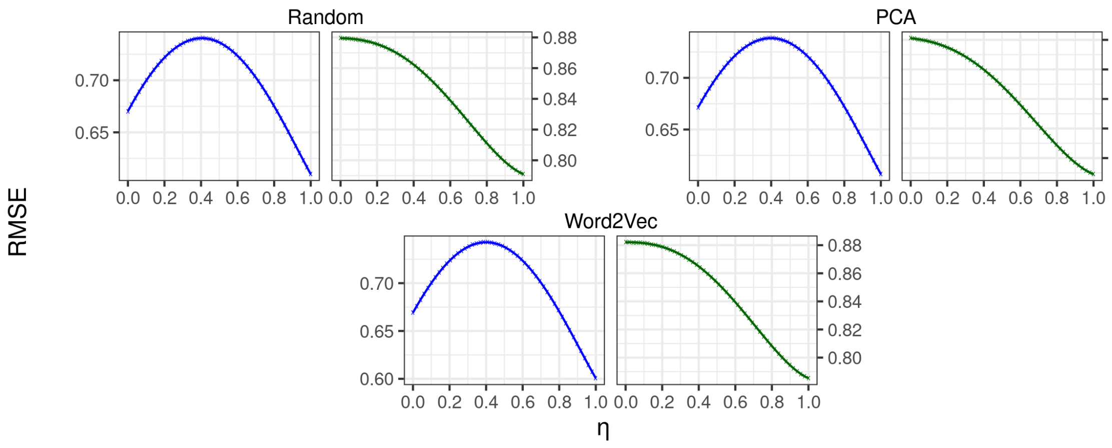

Figure B.12: Loss landscape analysis on a 100k sample for each initialization from MovieLens 24.9M in the *Source2Target* strategy.

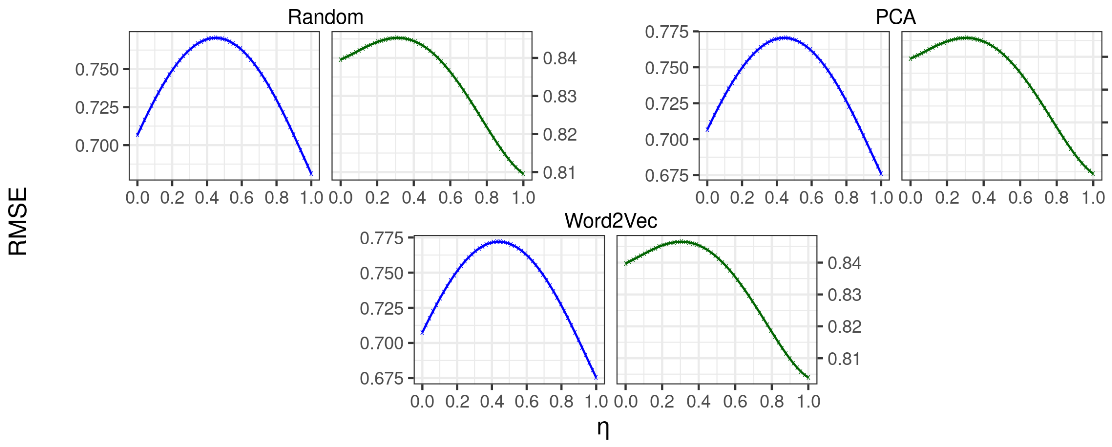

Figure B.13: Loss landscape analysis on a 1M sample for each initialization from MovieLens 9M in the *Source2Target* strategy.

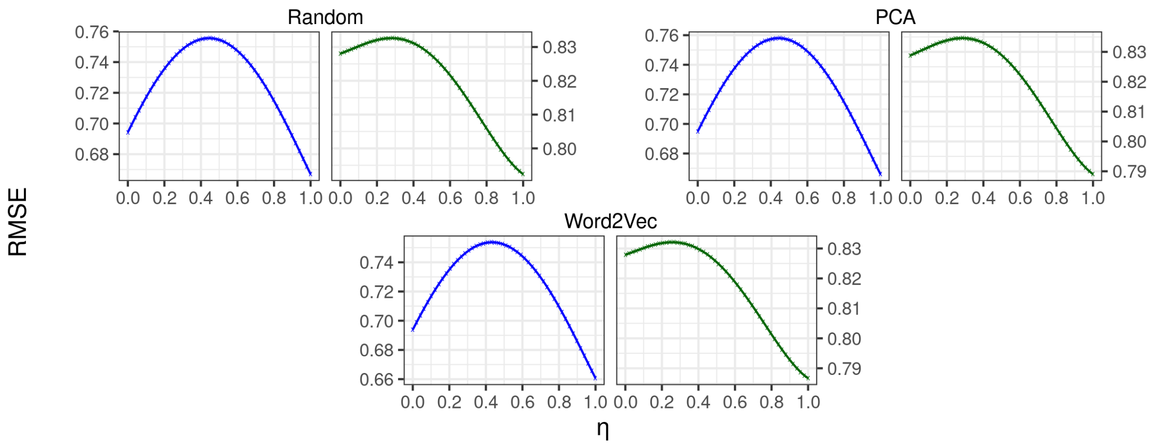

Figure B.14: Loss landscape analysis on a 1M sample for each initialization from MovieLens 19M in the *Source2Target* strategy.

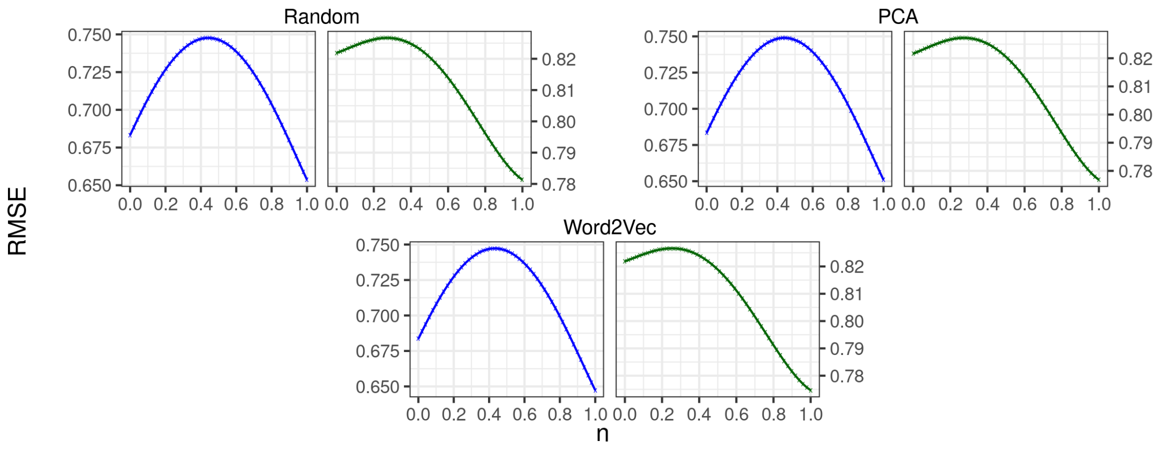

Figure B.15: Loss landscape analysis on a 1M sample for each initialization from MovieLens 24M in the *Source2Target* strategy.

# Appendix C

# Implicit Feedback Supplementary Material

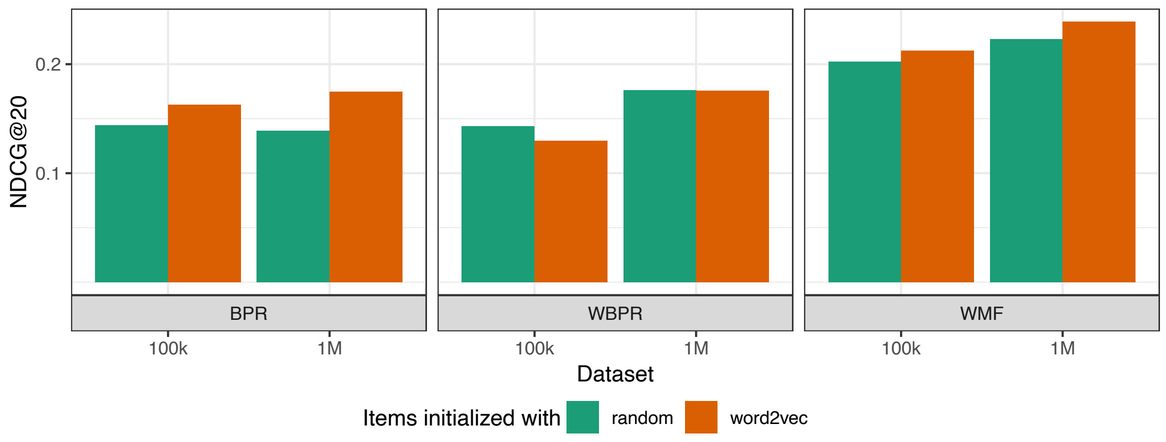

Figure C.1: Predictive performance regarding NDCG@20 on samples extracted from the MovieLens 25M dataset using the *Target2Target strategy*

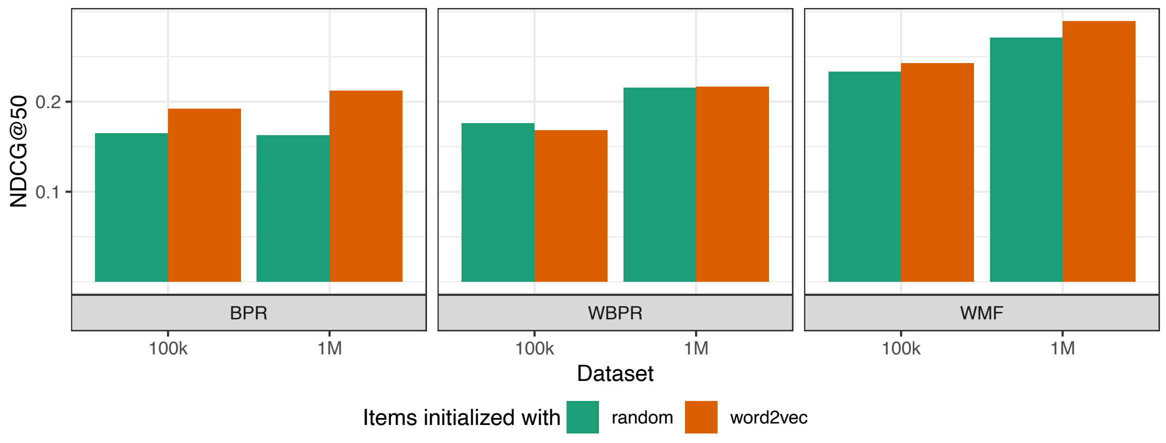

Figure C.2: Predictive performance regarding NDCG@50 on samples extracted from the MovieLens 25M dataset using the *Target2Target strategy*

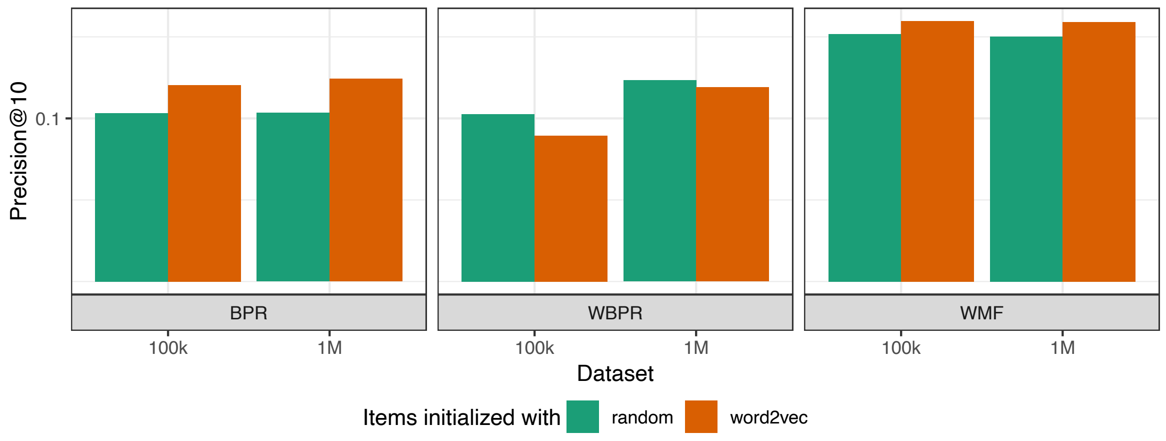

Figure C.3: Predictive performance regarding Precision@10 on samples extracted from the MovieLens 25M dataset using the *Target2Target strategy*

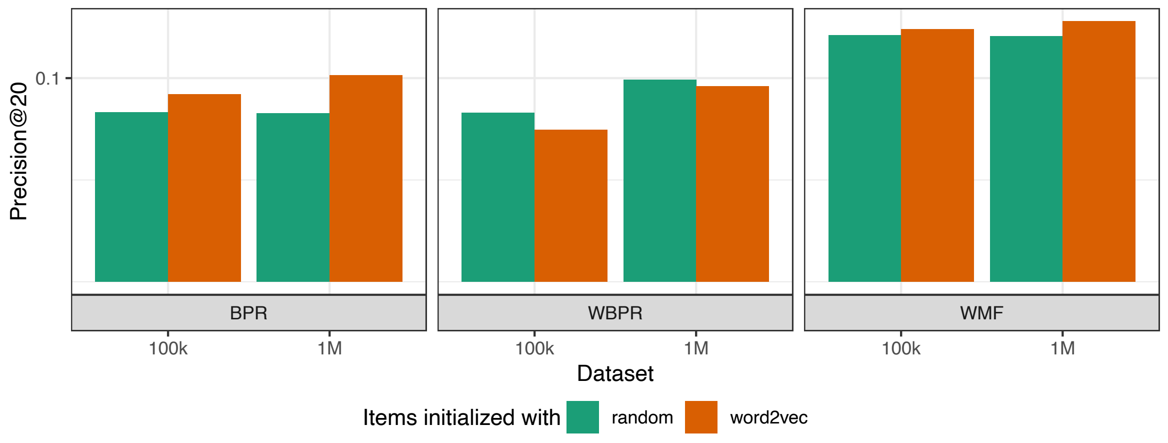

Figure C.4: Predictive performance regarding Precision@20 on samples extracted from the MovieLens 25M dataset using the *Target2Target strategy*

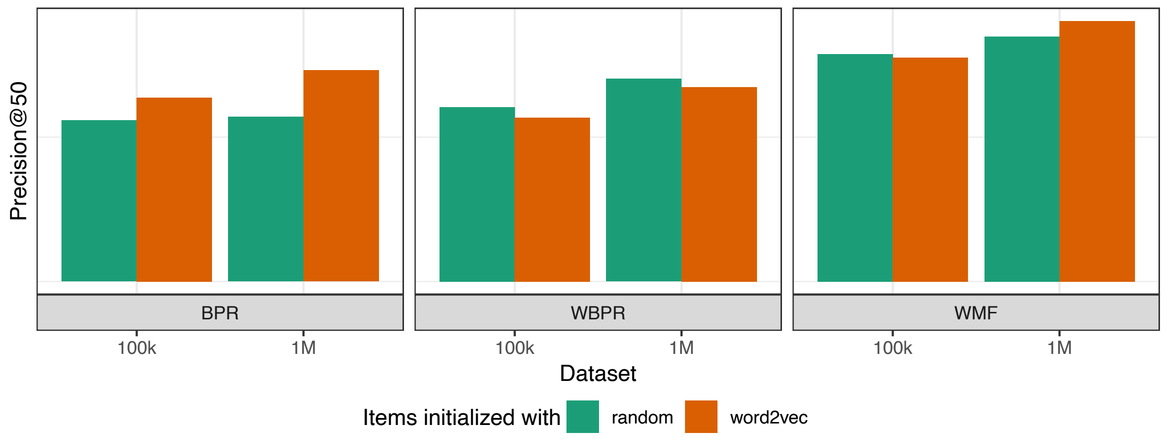

Figure C.5: Predictive performance regarding Precision@50 on samples extracted from the MovieLens 25M dataset using the *Target2Target strategy*

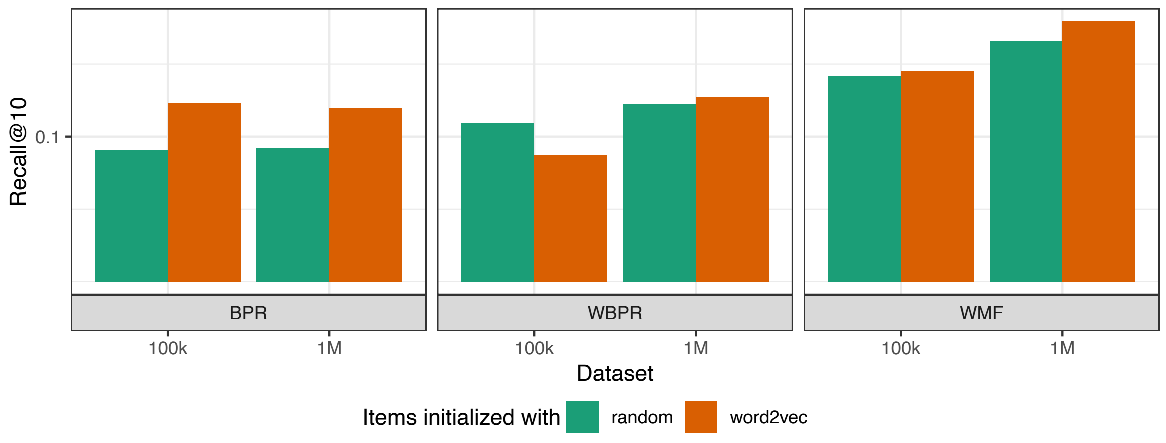

Figure C.6: Predictive performance regarding Recall@10 on samples extracted from the MovieLens 25M dataset using the *Target2Target strategy*

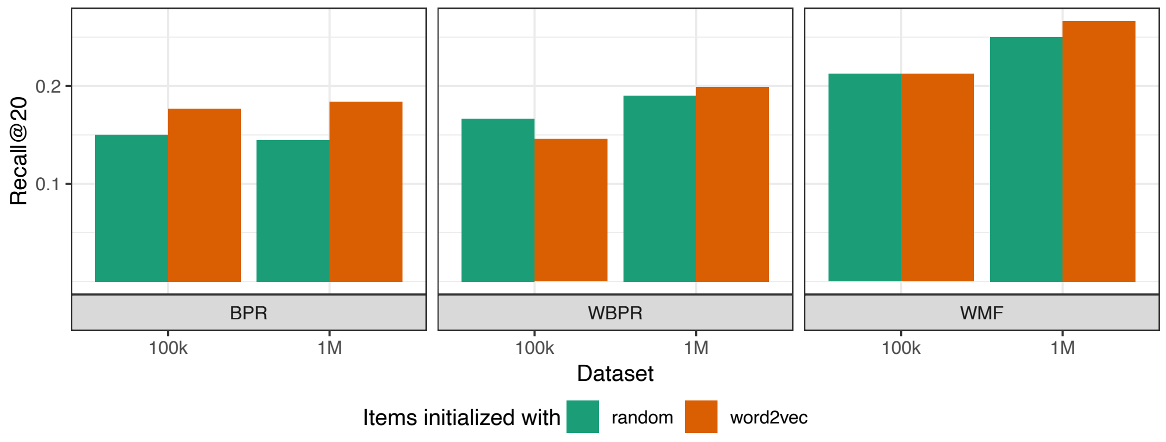

Figure C.7: Predictive performance regarding Recall@20 on samples extracted from the MovieLens 25M dataset using the *Target2Target strategy*

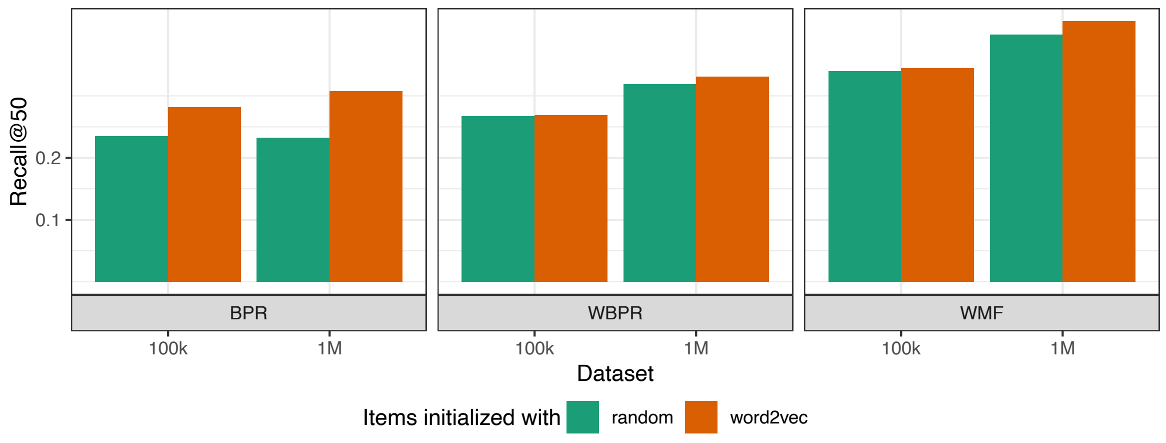

Figure C.8: Predictive performance regarding Recall@50 on samples extracted from the MovieLens 25M dataset using the *Target2Target strategy*

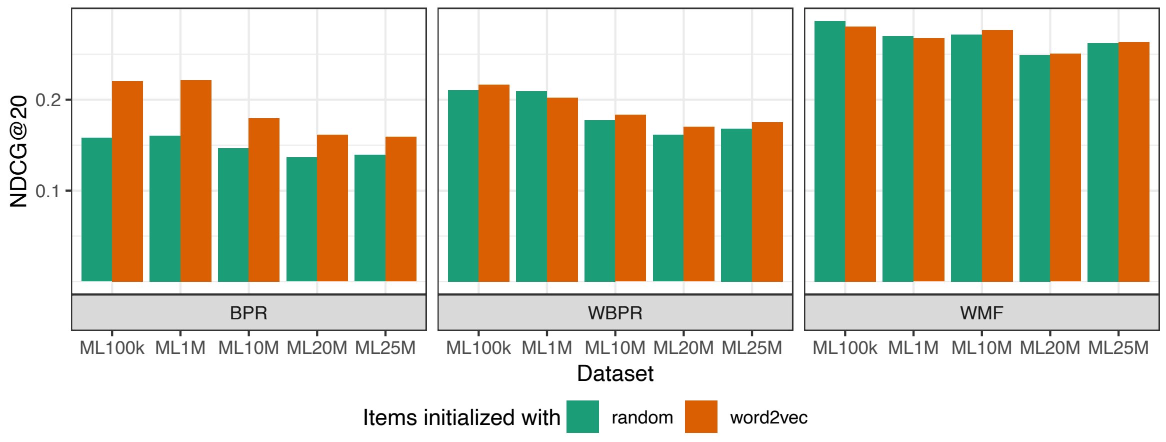

Figure C.9: Predictive performance in terms of NDCG@20 on the MovieLens datasets using the *Target2Target strategy*

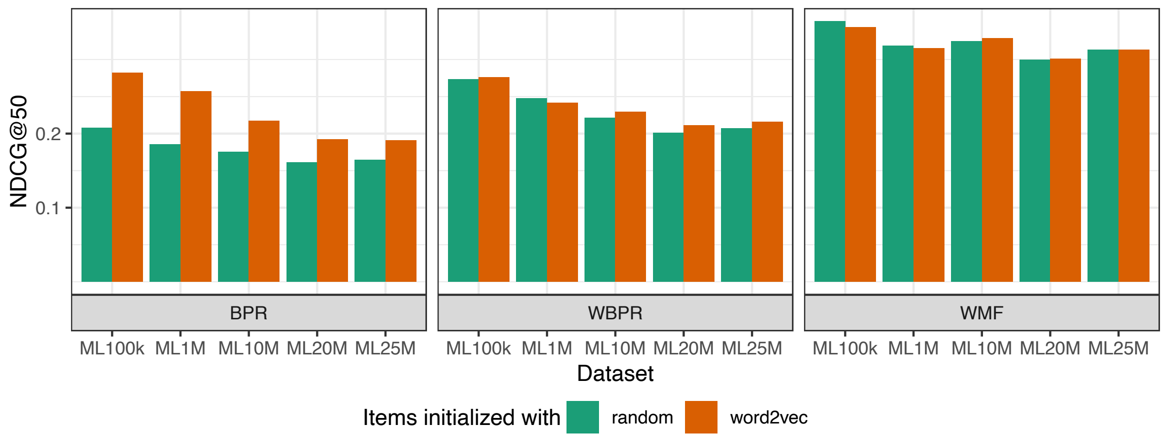

Figure C.10: Predictive performance in terms of NDCG@50 on the MovieLens datasets using the *Target2Target strategy*

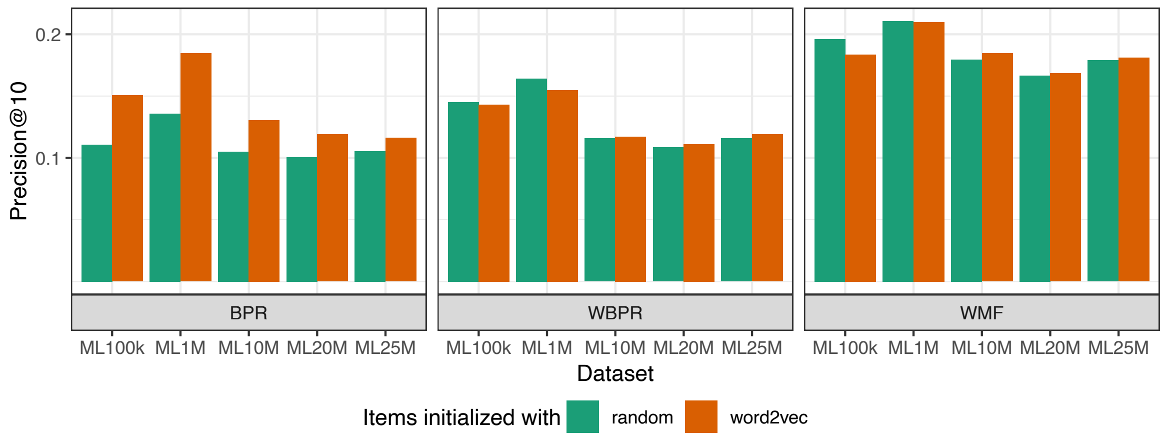

Figure C.11: Predictive performance in terms of Precision@10 on the MovieLens datasets using the *Target2Target strategy*

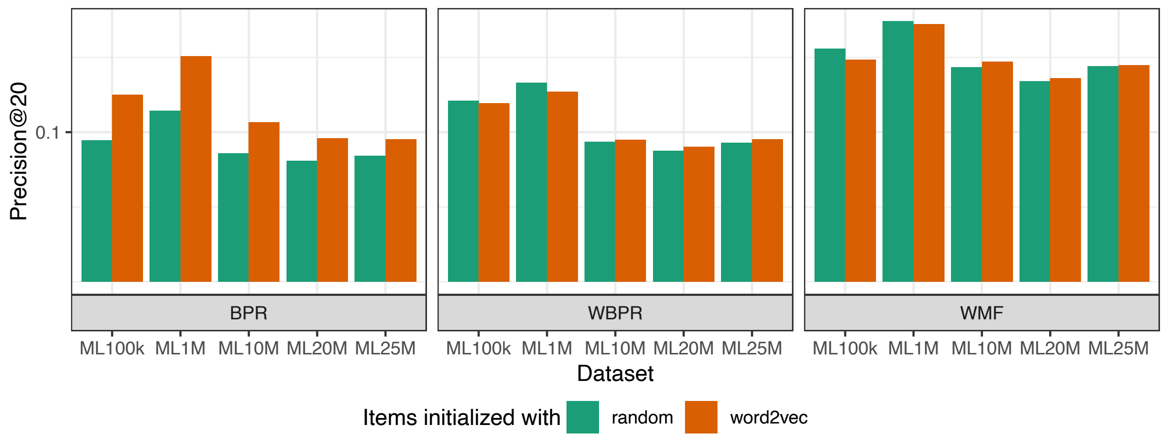

Figure C.12: Predictive performance in terms of Precision@20 on the MovieLens datasets using the *Target2Target strategy*

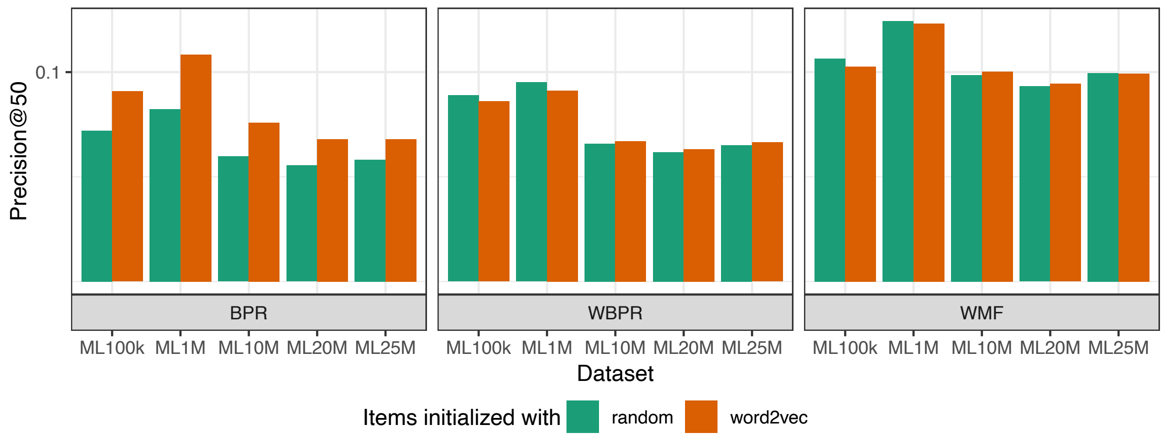

Figure C.13: Predictive performance in terms of Precision@50 on the MovieLens datasets using the *Target2Target strategy*

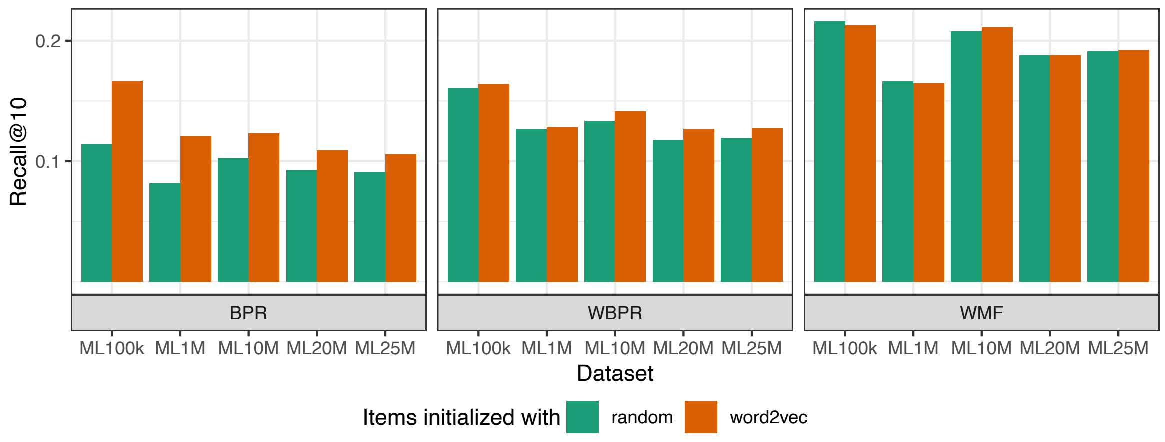

Figure C.14: Predictive performance in terms of Recall@10 on the MovieLens datasets using the *Target2Target strategy*

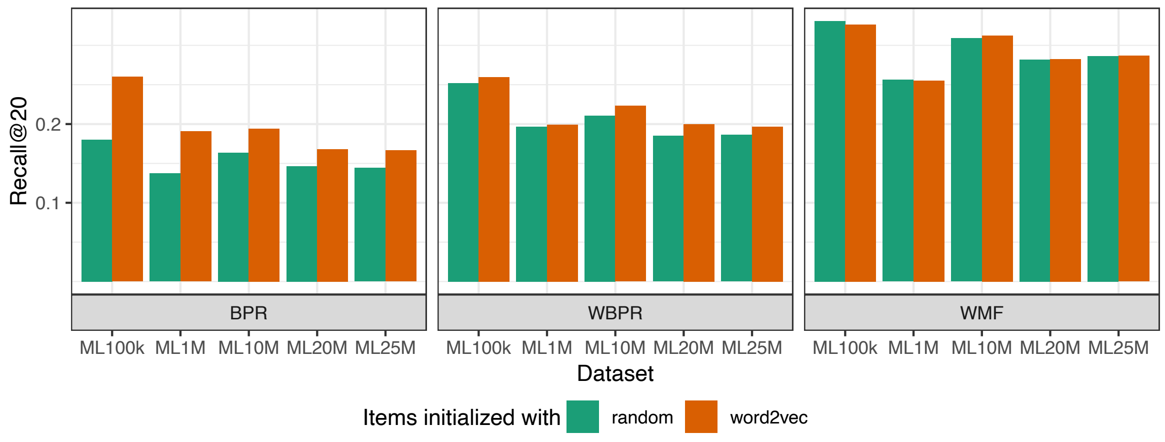

Figure C.15: Predictive performance in terms of Recall@20 on the MovieLens datasets using the *Target2Target strategy*

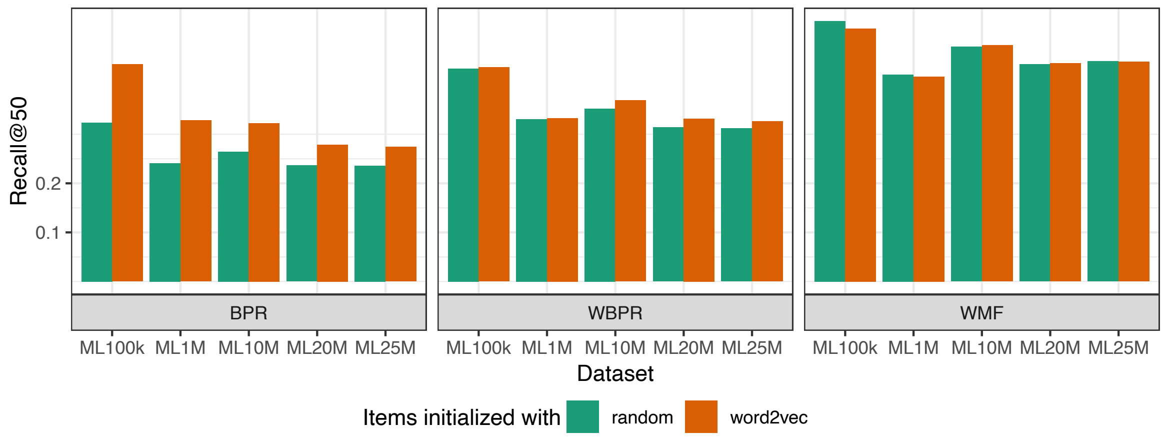

Figure C.16: Predictive performance in terms of Recall@50 on the MovieLens datasets using the *Target2Target strategy*

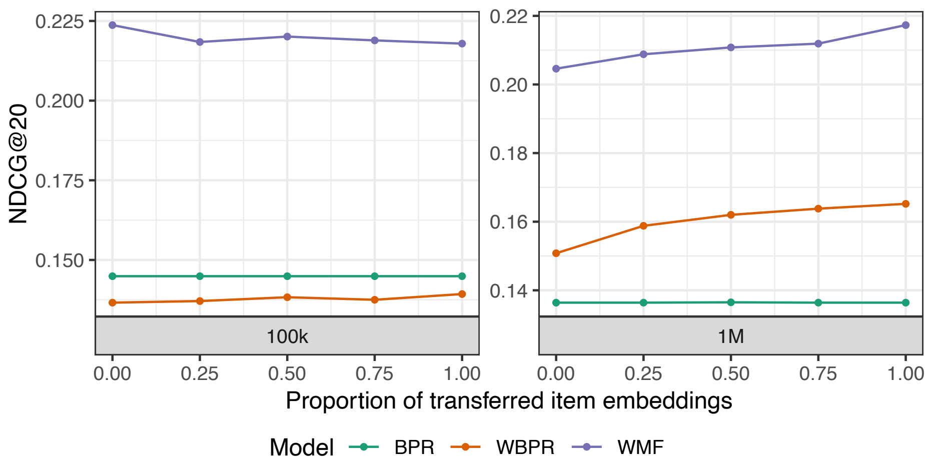

Figure C.17: Predictive performance regarding NDCG@20 on the 100k and 1M samples of the MovieLens 25M dataset using the *Source2Target strategy*

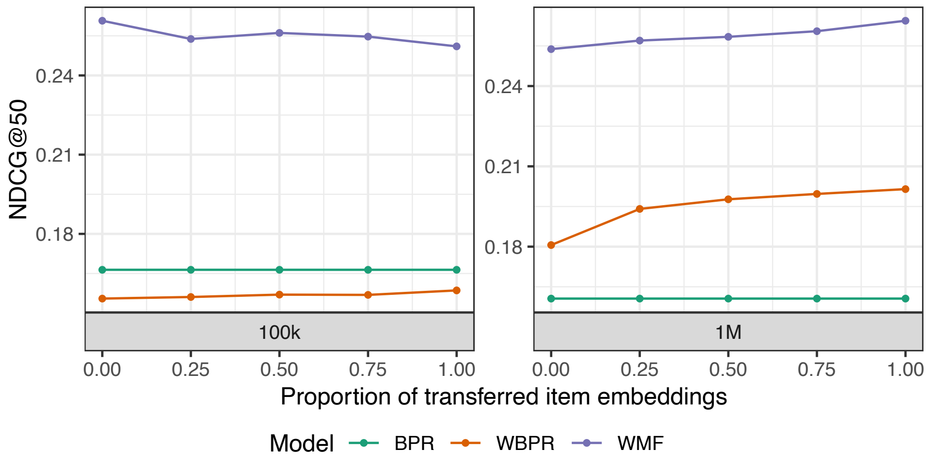

Figure C.18: Predictive performance regarding NDCG@50 on the 100k and 1M samples of the MovieLens 25M dataset using the *Source2Target strategy*

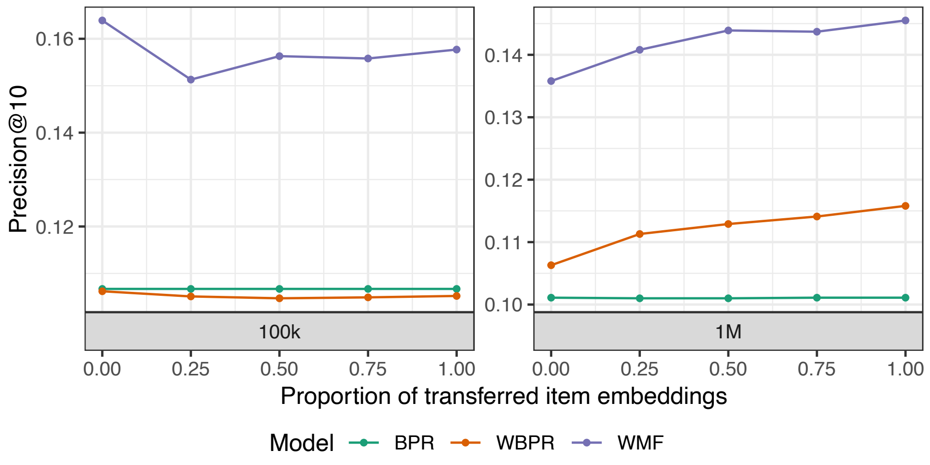

Figure C.19: Predictive performance regarding Precision@10 on the 100k and 1M samples of the MovieLens 25M dataset using the *Source2Target strategy*

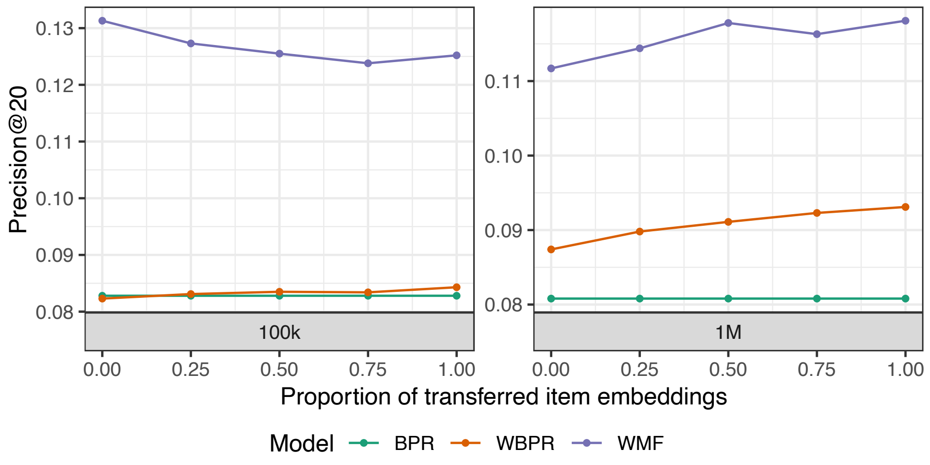

Figure C.20: Predictive performance regarding Precision@20 on the 100k and 1M samples of the MovieLens 25M dataset using the *Source2Target strategy*

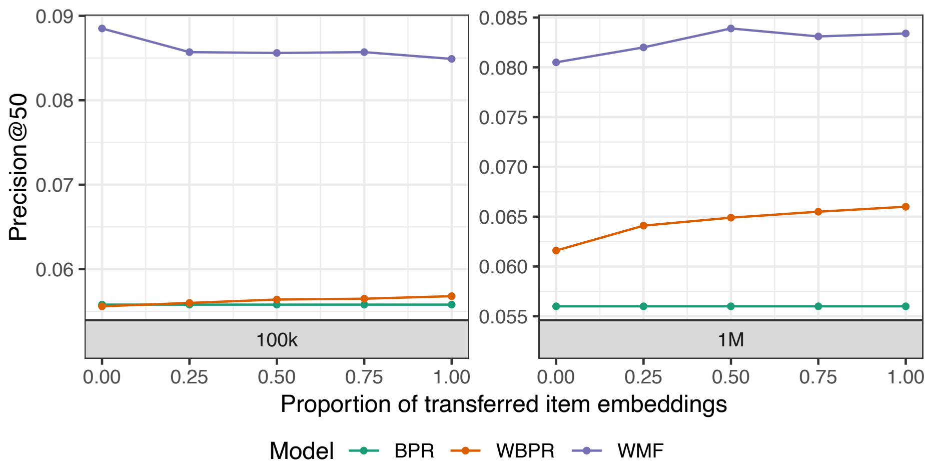

Figure C.21: Predictive performance regarding Precision@50 on the 100k and 1M samples of the MovieLens 25M dataset using the *Source2Target strategy*

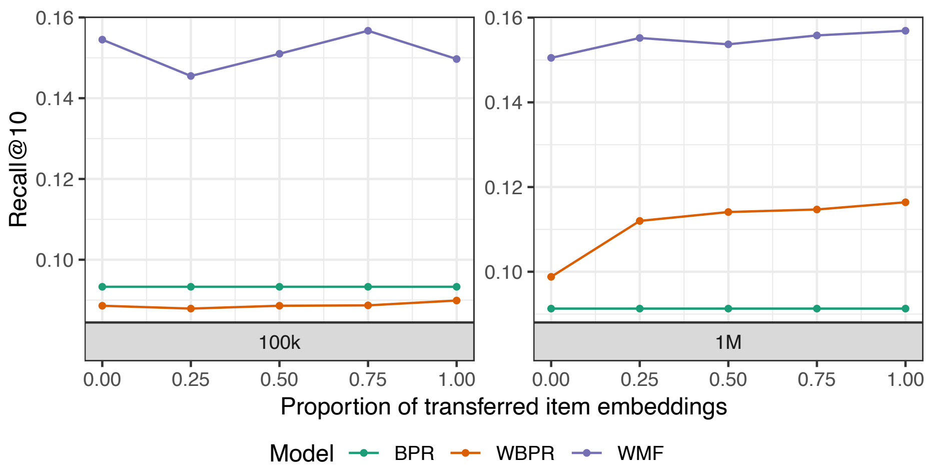

Figure C.22: Predictive performance regarding Recall@10 on the 100k and 1M samples of the MovieLens 25M dataset using the *Source2Target strategy*

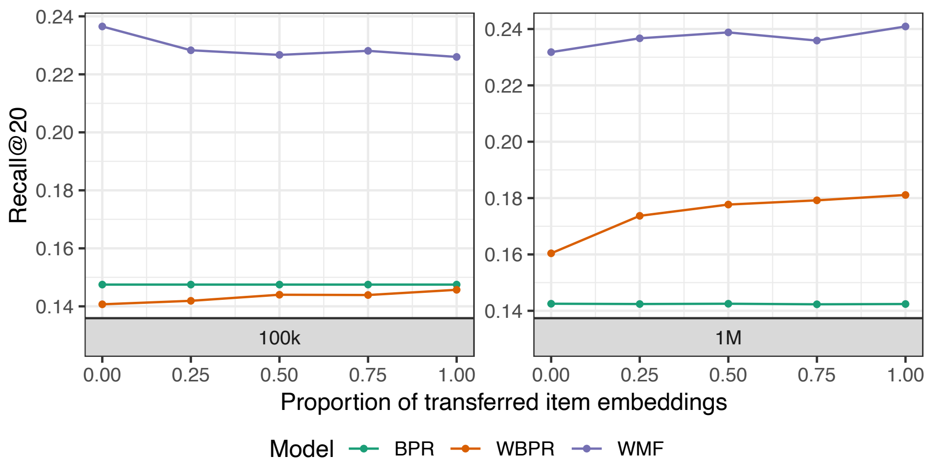

Figure C.23: Predictive performance regarding Recall@20 on the 100k and 1M samples of the MovieLens 25M dataset using the *Source2Target strategy*

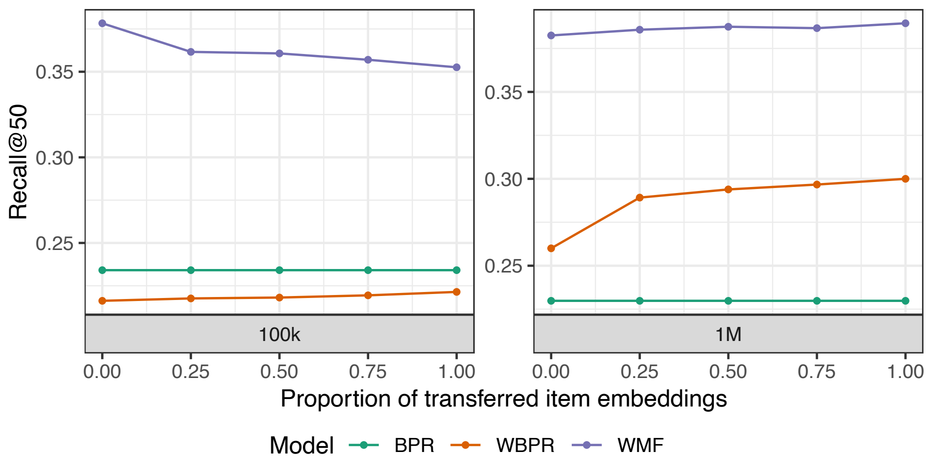

Figure C.24: Predictive performance regarding Recall@50 on the 100k and 1M samples of the MovieLens 25M dataset using the *Source2Target strategy*

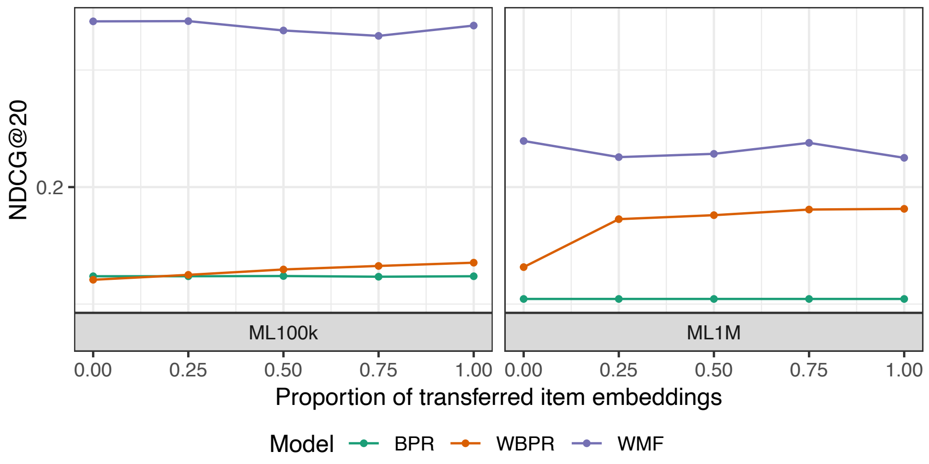

Figure C.25: Predictive performance in terms of NDCG@20 of models pre-trained with Netflix embeddings and evaluated in the MovieLens 100k and 1M datasets using the *Source2Target strategy*

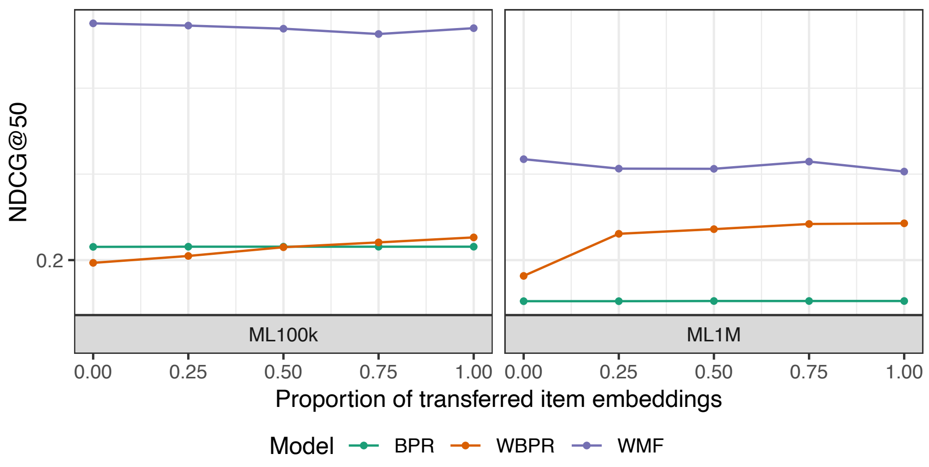

Figure C.26: Predictive performance in terms of NDCG@50 of models pre-trained with Netflix embeddings and evaluated in the MovieLens 100k and 1M datasets using the *Source2Target strategy*

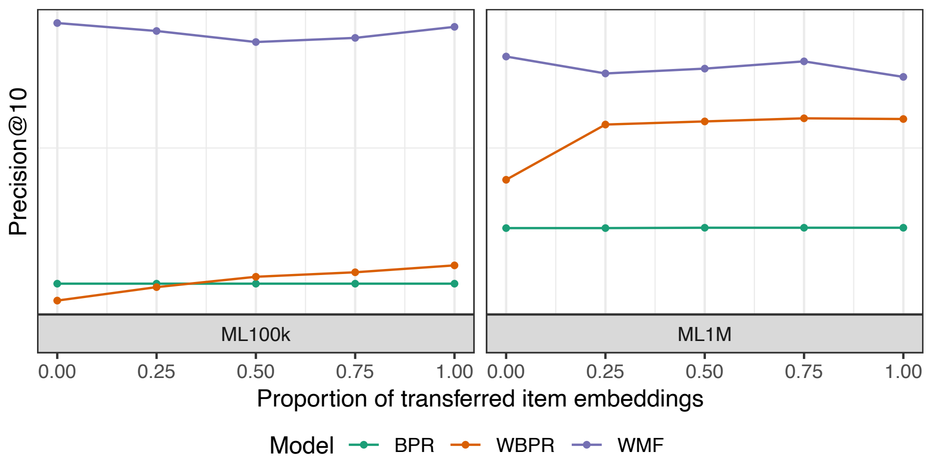

Figure C.27: Predictive performance in terms of Precision@10 of models pre-trained with Netflix embeddings and evaluated in the MovieLens 100k and 1M datasets using the *Source2Target strategy*

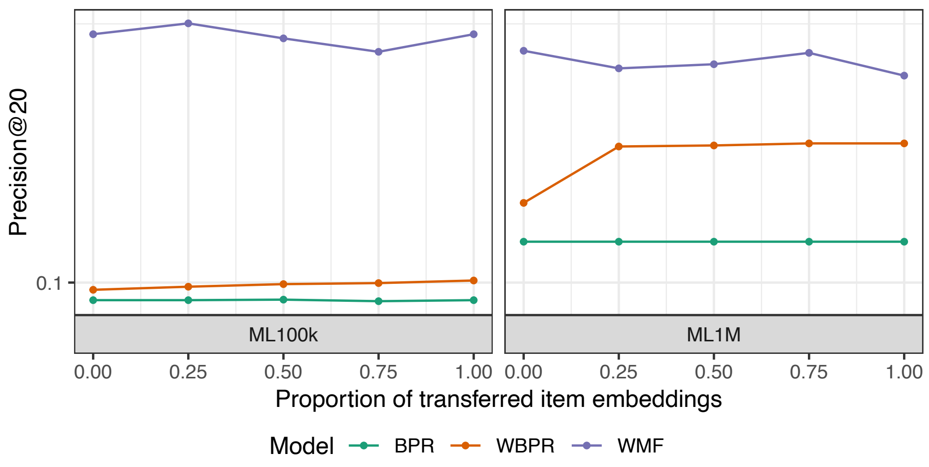

Figure C.28: Predictive performance in terms of Precision@20 of models pre-trained with Netflix embeddings and evaluated in the MovieLens 100k and 1M datasets using the *Source2Target strategy*

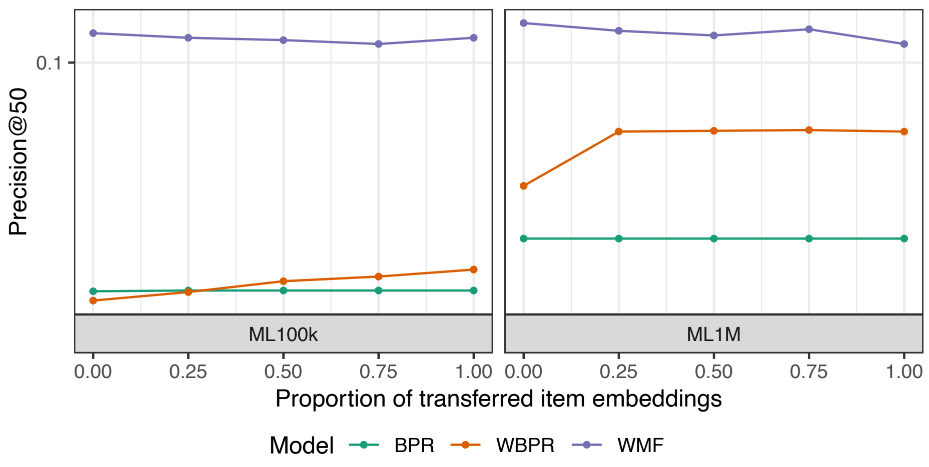

Figure C.29: Predictive performance in terms of Precision@50 of models pre-trained with Netflix embeddings and evaluated in the MovieLens 100k and 1M datasets using the *Source2Target strategy*

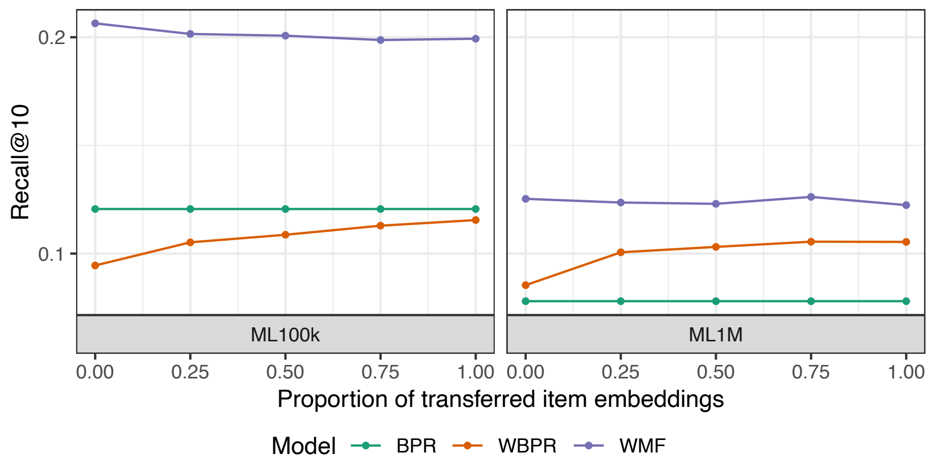

Figure C.30: Predictive performance in terms of Recall@10 of models pre-trained with Netflix embeddings and evaluated in the MovieLens 100k and 1M datasets using the *Source2Target strategy*

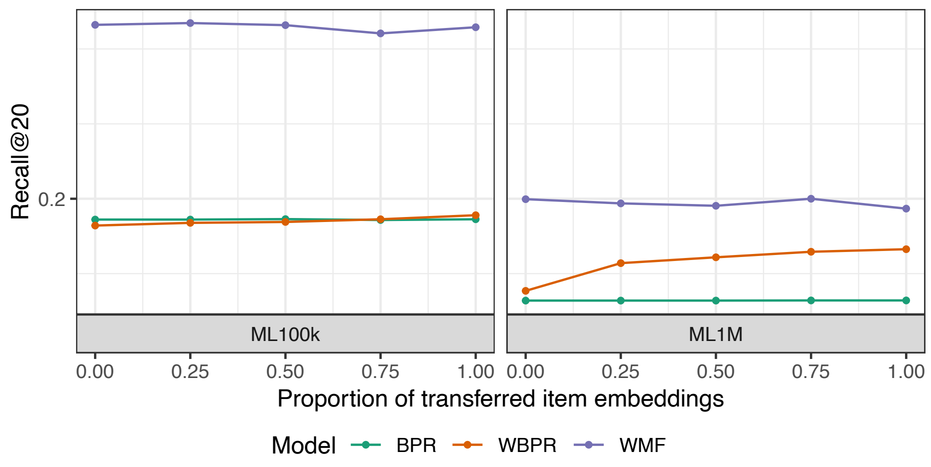

Figure C.31: Predictive performance in terms of Recall@20 of models pre-trained with Netflix embeddings and evaluated in the MovieLens 100k and 1M datasets using the *Source2Target strategy*

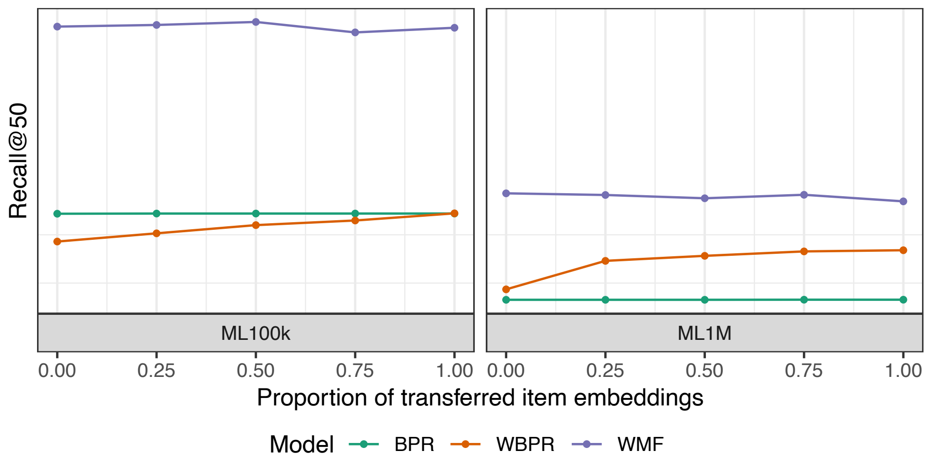

Figure C.32: Predictive performance in terms of Recall@50 of models pre-trained with Netflix embeddings and evaluated in the MovieLens 100k and 1M datasets using the *Source2Target strategy*

# Appendix D

# Cross-task Supplementary Material

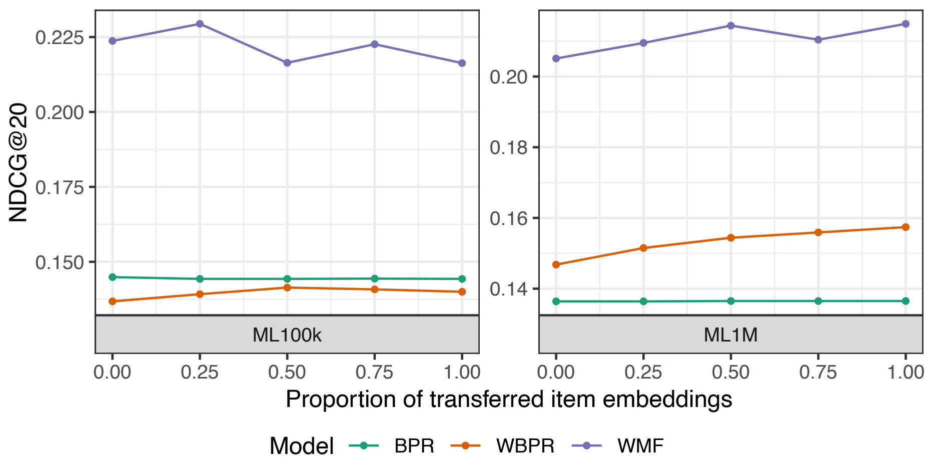

Figure D.1: Predictive performance in terms of NDCG@20 on the $100$k and $1$M samples of the MovieLens $25$M dataset using the *Rating2Ranking strategy*

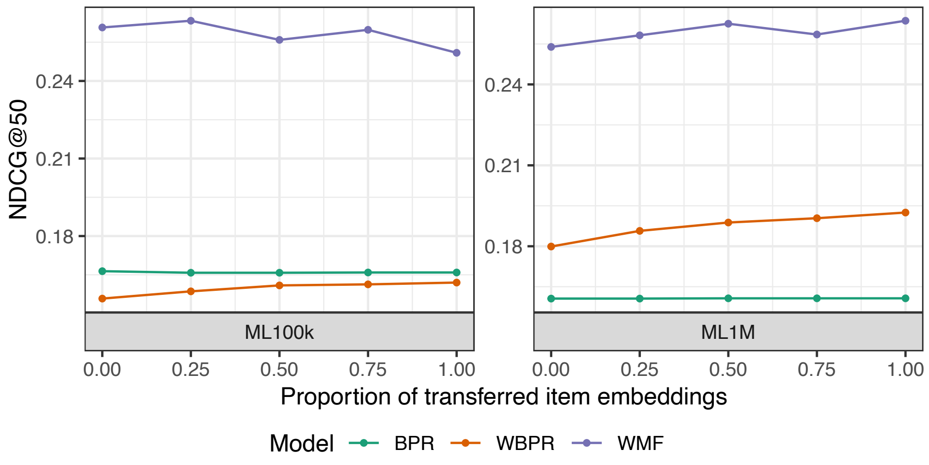

Figure D.2: Predictive performance in terms of NDCG@50 on the 100k and 1M samples of the MovieLens 25M dataset using the *Rating2Ranking strategy*

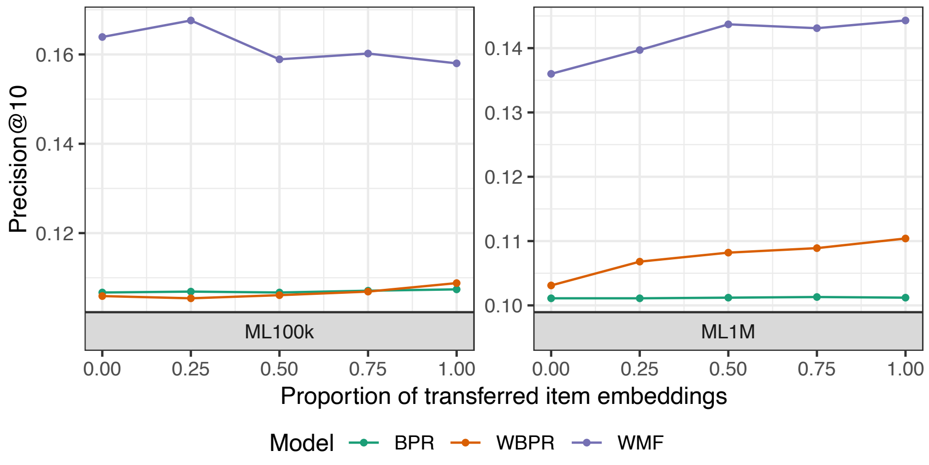

Figure D.3: Predictive performance in terms of Precision@10 on the 100k and 1M samples of the MovieLens 25M dataset using the *Rating2Ranking strategy*

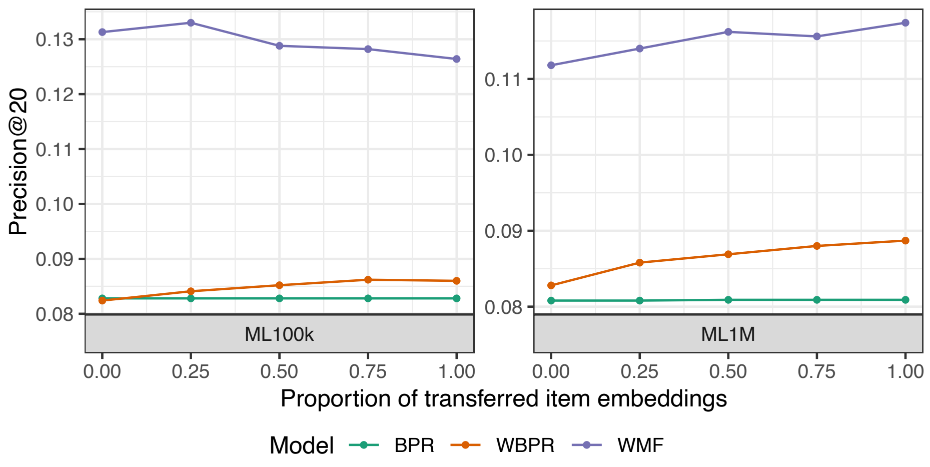

Figure D.4: Predictive performance in terms of Precision@20 on the 100k and 1M samples of the MovieLens 25M dataset using the *Rating2Ranking strategy*

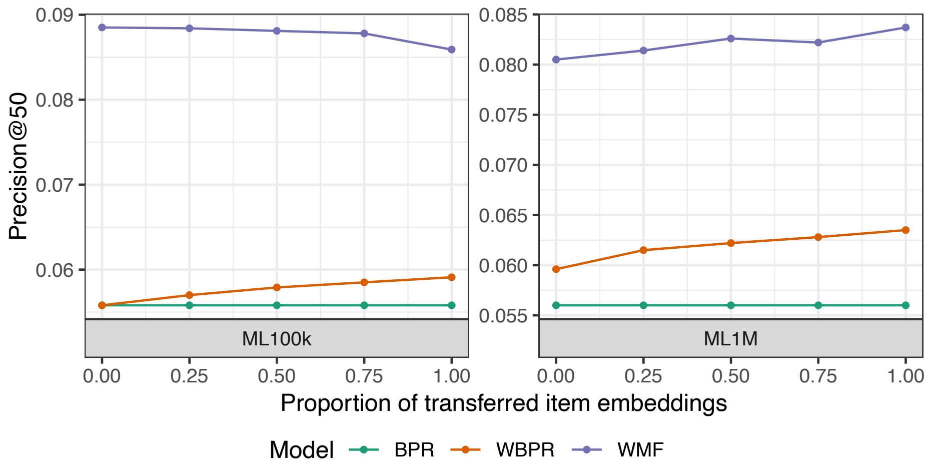

Figure D.5: Predictive performance in terms of Precision@50 on the 100k and 1M samples of the MovieLens 25M dataset using the *Rating2Ranking strategy*

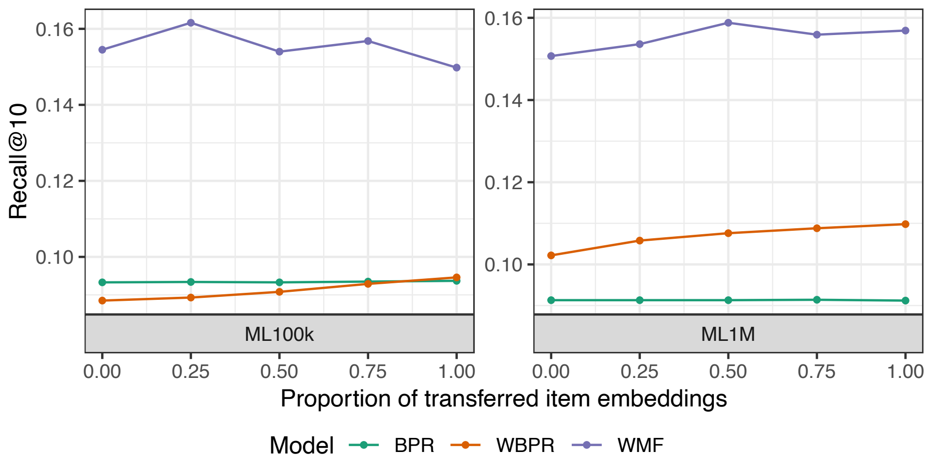

Figure D.6: Predictive performance in terms of Recall@10 on the 100k and 1M samples of the MovieLens 25M dataset using the *Rating2Ranking strategy*

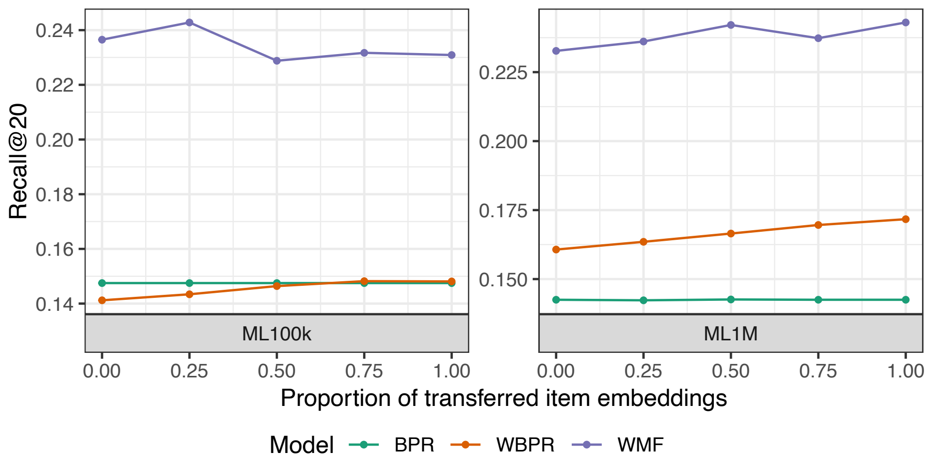

Figure D.7: Predictive performance in terms of Recall@20 on the 100k and 1M samples of the MovieLens 25M dataset using the *Rating2Ranking strategy*

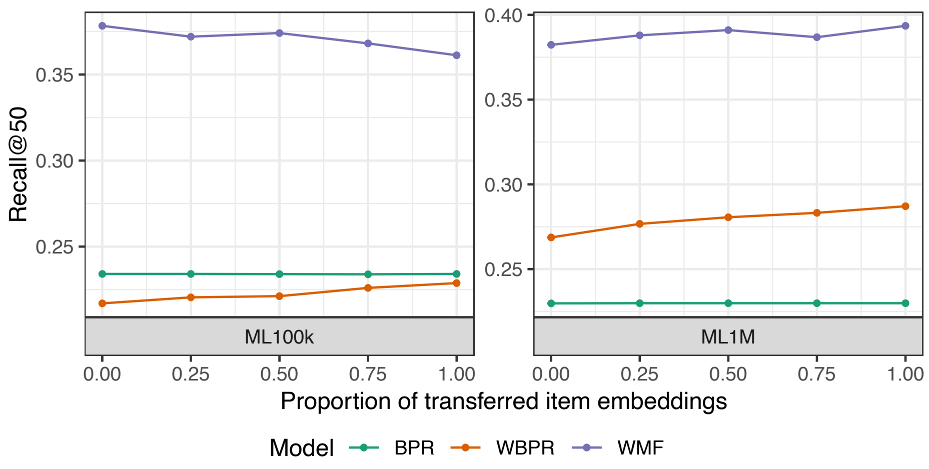

Figure D.8: Predictive performance in terms of Recall@50 on the 100k and 1M samples of the MovieLens 25M dataset using the *Rating2Ranking strategy*

# Bibliography

[1] Hervé Abdi and Lynne J. Williams. Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459, 2010.

[2] ACM SIGKDD. KDD Cup and Workshop 2007. `https://www.cs.uic.edu/~liub/Netflix-KDD-Cup-2007.html#submission`.

[3] Yilmaz Ar. An initialization method for the latent vectors in probabilistic matrix factorization for sparse datasets. *Evolutionary Intelligence*, pages 1–13, 2019.

[4] Max Bachmann. maxbachmann/rapidfuzz: Release 1.8.0, October 2021.

[5] Oren Barkan and Noam Koenigstein. Item2vec: neural item embedding for collaborative filtering. In *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2016.

[6] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. *Semantics-Aware Content-Based Recommender Systems*, pages 119–159. Springer US, Boston, MA, 2015.

[7] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. Transformers4rec: Bridging the gap between nlp and sequential / session-based recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems*, RecSys '21, page 143–153, New York, NY, USA, 2021. Association for Computing Machinery.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[9] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council.

[10] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, page 101–109, New York, NY, USA, 2019. Association for Computing Machinery.

[11] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. Personalized ranking for non-uniformly sampled items. In *Proceedings of KDD Cup 2011*, pages 231–247. PMLR, 2012.

[12] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[13] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[14] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1809–1818, 2015.

[15] Soyeon Caren Han, Taejun Lim, Siqu Long, Bernd Burgstaller, and Josiah Poon. Glocal-k: Global and local kernels for recommender systems. In *CIKM 2021 - Proceedings of the 30th ACM International Conference on Information and Knowledge Management*, International Conference on Information and Knowledge Management, Proceedings, pages 3063–3067. Association for Computing Machinery, October 2021. Publisher Copyright: © 2021 ACM.; 30th ACM International Conference on Information and Knowledge Management, CIKM 2021 ; Conference date: 01-11-2021 Through 05-11-2021.

[16] Casper Hansen, Christian Hansen, Lucas Maystre, Rishabh Mehrotra, Brian Brost, Federico Tomasi, and Mounia Lalmas. Contextual and sequential user embeddings for large-scale music recommendation. In *Fourteenth ACM Conference on Recommender Systems*, page 53–62, New York, NY, USA, 2020. Association for Computing Machinery.

[17] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015.

[18] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[20] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 173–182, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[21] Balázs Hidasi and Domonkos Tikk. Enhancing matrix factorization through initialization for implicit feedback databases. In *Proceedings of the 2nd Workshop on Context-awareness in Retrieval and Recommendation*, pages 2–9, 2012.

[22] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, Oct 2018.

[23] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks, 2016.

[24] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272, 2008.

[25] Nicolas Hug. Surprise: A python library for recommender systems. *Journal of Open Source Software*, 5(52):2174, 2020.

[26] Matthew A. Jaro. Probabilistic linkage of large public health data files. *Statistics in Medicine*, 14(5-7):491–498, 1995.

[27] Olivier Jeunen, Jan Van Balen, and Bart Goethals. *Closed-Form Models for Collaborative Filtering with Side-Information*, page 651–656. Association for Computing Machinery, New York, NY, USA, 2020.

[28] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[29] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[31] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[32] Yan Leng, Rodrigo Ruiz, and Xiao Liu. Geometric deep learning based recommender system and an interpretable decision support system. *Available at SSRN 3696092*, 2020.

[33] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

[34] Xiaopeng Li and James She. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 305–314, New York, NY, USA, 2017. Association for Computing Machinery.

[35] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM conference on recommender systems*, pages 59–66, 2016.

[36] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, page 689–698, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee.

[37] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[38] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N. Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*, pages 502–511, 2008.

[39] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.

[40] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, 2018.

[41] Aleksandr Petrov and Craig Macdonald. A systematic review and replicability study of bert4rec for sequential recommendation. In *Proceedings of the 16th ACM Conference on Recommender Systems*, RecSys '22, page 436–447, New York, NY, USA, 2022. Association for Computing Machinery.

[42] "Simon J.D. Prince". *"Understanding Deep Learning"*. "MIT Press", 2023.

[43] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[44] Ahmed Rashed, Josif Grabocka, and Lars Schmidt-Thieme. Attribute-aware non-linear co-embeddings of graph features. RecSys '19, page 314–321, New York, NY, USA, 2019. Association for Computing Machinery.

[45] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[46] Steffen Rendle, Walid Krichene, Li Zhang, and Yehuda Koren. Revisiting the performance of ials on item recommendation benchmarks. In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 427–435, 2022.

[47] Steffen Rendle, Li Zhang, and Yehuda Koren. On the difficulty of evaluating baselines: A study on recommender systems, 2019.

[48] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.

[49] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[50] Aghiles Salah, Quoc-Tuan Truong, and Hady W Lauw. Cornac: A comparative framework for multimodal recommender systems. *Journal of Machine Learning Research*, 21(95):1–5, 2020.

[51] Mathias Seuret, Michele Alberti, Marcus Liwicki, and Rolf Ingold. Pca-initialized deep neural networks applied to document image analysis. In *2017 14th IAPR international conference on document analysis and recognition (ICDAR)*, volume 1, pages 877–882. IEEE, 2017.

[52] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014.

[53] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, and Sergey I Nikolenko. Recvae: A new variational autoencoder for top-n recommendations with

implicit feedback. In *Proceedings of the 13th international conference on web search and data mining*, pages 528–536, 2020.

[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[55] Giuseppe Spillo, Cataldo Musto, Marco Polignano, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Combining graph neural networks and sentence encoders for knowledge-aware recommendations. In *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '23, page 1–12, New York, NY, USA, 2023. Association for Computing Machinery.

[56] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1441–1450, New York, NY, USA, 2019. Association for Computing Machinery.

[57] Vojtěch Vančura and Pavel Kordík. Deep variational autoencoder with shallow parallel path for top-n recommendation (vasp). In *International Conference on Artificial Neural Networks*, pages 138–149. Springer, 2021.

[58] Robin Verachtert, Lien Michiels, and Bart Goethals. Are we forgetting something? correctly evaluate a recommender system with an optimal training window. In *Proceedings of the Perspectives on the Evaluation of Recommender Systems Workshop*, volume 3228, 2022.

[59] Chen Wang, Yueqing Liang, Zhiwei Liu, Tao Zhang, and Philip S. Yu. Pre-training graph neural network for cross domain recommendation. In *2021 IEEE Third International Conference on Cognitive Machine Intelligence (CogMI)*, pages 140–145, 2021.

[60] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3:1–40, 2016.

[61] William E Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. 1990.

[62] William E Winkler. Overview of record linkage and current research directions. In *Bureau of the Census*. Citeseer, 2006.

[63] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[64] Liwei Wu, Shuqing Li, Cho-Jui Hsieh, and James Sharpnack. Sse-pt: Sequential recommendation via personalized transformer. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 328–337, 2020.

[65] Sun Wu and Udi Manber. Fast text searching: Allowing errors. *Commun. ACM*, 35(10):83–91, oct 1992.

[66] Bereket A. Yilma and Luis A. Leiva. Together yet apart: Multimodal representation learning for personalised visual art recommendation. In *Proceedings of the 31st ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '23, page 204–214, New York, NY, USA, 2023. Association for Computing Machinery.

[67] Zahra Zamanzadeh Darban and Mohammad Hadi Valipour. Ghrs: Graph-based hybrid recommendation system with application to movie recommendation. *Expert Systems with Applications*, 200:116850, 2022.

[68] Yin Zhang, Ziwei Zhu, Yun He, and James Caverlee. *Content-Collaborative Disentanglement Representation Learning for Enhanced Recommendation*, page 43–52. Association for Computing Machinery, New York, NY, USA, 2020.

[69] Wayne Xin Zhao, Zihan Lin, Zhichao Feng, Pengfei Wang, and Ji-Rong Wen. A revisiting study of appropriate offline evaluation for top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 41(2), dec 2022.