Cross-domain recommender systems using deep neural networks

by

Nawaf Alharbi

MS, California State University - Long Beach, 2016

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2021

Abstract

With the continuous growth in the number of products available in e-commerce applications and information items available on the Internet, the task of associating users with a small list of personalized items, extracted from a large and diverse pool of items, is clearly beyond human ability, a problem known as *Information Overload*. Indeed, the task (or ability) of automatically, quickly, and accurately recommending appropriate items to users has become an essential part in determining the success of almost all e-commerce businesses and online service providers. Recommender Systems (RS) aim to exploit users' historical interaction records, to capture users' preferences, and accordingly identify items that may be of interest to particular users. Recommender systems have become an integral part of our lives. People interact with recommender systems on a daily basis for leisure activities, such as finding a movie to watch, a friend to follow, etc., or for professional activities, such as finding a topic to study, finding co-authors to collaborate with, etc. Furthermore, online service providers rely heavily on recommender systems to increase their revenue, to diversify their item recommendations (and indirectly sales), and ultimately to keep their customers satisfied and engaged. Without recommender systems, it seems impossible for online businesses to survive in this competitive world-market.

Most available recommender systems are focused on single-domain recommendation tasks (e.g., the task of recommending books to particular users, based on the book history of a large number of users). However, for users with a limited behavior history (i.e., a small number of interactions), single-domain recommender systems perform poorly, suffering from the data sparsity and cold start problems. Recent developments in the general area of *transfer learning* have led to Cross-Domain Recommender Systems, which exploit user behaviour information from other source domains and transfer this information to a target domain. This knowledge transfer helps alleviate the sparsity issue and produce more accurate recommendations in

the target domain. This is possible because users generally interact with multiple, related domains in their daily lives.

In this dissertation, I focus on cross-domain recommender system approaches, which employ state-of-the-art deep neural networks, to tackle the data sparsity problem in a target domain. The source domains used to gain additional information in the cross-domain setting are selected to have user overlap with the target domain. Specifically, I propose three novel cross-domain recommendation models, which leverage state-of-the-art single-task recommendation models. The first proposed cross-domain model is a non-sequential recommender system (i.e., the order of the items in a user history is not considered), and it uses a neural collaborative filtering approach to performs knowledge transfer between the source and target at the embedding layer level. The second proposed approach is focused on sequential recommendations, and uses the successful self-attention mechanism to identify useful item preferences in both source and target domains. It also uses the early fusion technique to combine a pre-learned global source representation of a user with a target representation of the user. Finally, the third proposed cross-domain model uses one or more source domains to obtain users' general preferences, while the target domain is used to extract users' current preferences. The two types of preferences, general and target-specific preferences expressed as representational vectors, are then fused together to achieve knowledge transfer across domains and improve the recommendation accuracy in the target domain, especially when the target domain faces the sparsity problem. Public cross-domain datasets are employed to evaluate our proposed models in different settings relevant to target sparsity. Experimental results show that our proposed models increase the recommendation accuracy in the target domain, outperforming existing state-of-the-art recommender system models.

Cross-domain recommender systems using deep neural networks

by

Nawaf Alharbi

MS, California State University - Long Beach, 2016

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2021

Approved by:

Major Professor
Doina Caragea

Copyright

Abstract

With the continuous growth in the number of products available in e-commerce applications and information items available on the Internet, the task of associating users with a small list of personalized items, extracted from a large and diverse pool of items, is clearly beyond human ability, a problem known as *Information Overload*. Indeed, the task (or ability) of automatically, quickly, and accurately recommending appropriate items to users has become an essential part in determining the success of almost all e-commerce businesses and online service providers. Recommender Systems (RS) aim to exploit users' historical interaction records, to capture users' preferences, and accordingly identify items that may be of interest to particular users. Recommender systems have become an integral part of our lives. People interact with recommender systems on a daily basis for leisure activities, such as finding a movie to watch, a friend to follow, etc., or for professional activities, such as finding a topic to study, finding co-authors to collaborate with, etc. Furthermore, online service providers rely heavily on recommender systems to increase their revenue, to diversify their item recommendations (and indirectly sales), and ultimately to keep their customers satisfied and engaged. Without recommender systems, it seems impossible for online businesses to survive in this competitive world-market.

Most available recommender systems are focused on single-domain recommendation tasks (e.g., the task of recommending books to particular users, based on the book history of a large number of users). However, for users with a limited behavior history (i.e., a small number of interactions), single-domain recommender systems perform poorly, suffering from the data sparsity and cold start problems. Recent developments in the general area of *transfer learning* have led to Cross-Domain Recommender Systems, which exploit user behaviour information from other source domains and transfer this information to a target domain. This knowledge transfer helps alleviate the sparsity issue and produce more accurate recommendations in

the target domain. This is possible because users generally interact with multiple, related domains in their daily lives.

In this dissertation, I focus on cross-domain recommender system approaches, which employ state-of-the-art deep neural networks, to tackle the data sparsity problem in a target domain. The source domains used to gain additional information in the cross-domain setting are selected to have user overlap with the target domain. Specifically, I propose three novel cross-domain recommendation models, which leverage state-of-the-art single-task recommendation models. The first proposed cross-domain model is a non-sequential recommender system (i.e., the order of the items in a user history is not considered), and it uses a neural collaborative filtering approach to performs knowledge transfer between the source and target at the embedding layer level. The second proposed approach is focused on sequential recommendations, and uses the successful self-attention mechanism to identify useful item preferences in both source and target domains. It also uses the early fusion technique to combine a pre-learned global source representation of a user with a target representation of the user. Finally, the third proposed cross-domain model uses one or more source domains to obtain users' general preferences, while the target domain is used to extract users' current preferences. The two types of preferences, general and target-specific preferences expressed as representational vectors, are then fused together to achieve knowledge transfer across domains and improve the recommendation accuracy in the target domain, especially when the target domain faces the sparsity problem. Public cross-domain datasets are employed to evaluate our proposed models in different settings relevant to target sparsity. Experimental results show that our proposed models increase the recommendation accuracy in the target domain, outperforming existing state-of-the-art recommender system models.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my deepest gratitude to my advisor Dr. Doina Caragea for her continuous support and valuable advice during my Ph.D study and research. Over these years, I have learned a lot from her, not only the research skills, but also how to conquer research obstacles. Her continuous encouragement has helped me keep moving forward from the very beginning of this long ongoing journey. Without her assistance, this dissertation would not have been possible.

I would also like to thank my committee members Dr. Torben Amtoft, Dr. Dan Andresen, and Dr. Nathan Albin, for their participation and the constructive suggestions for the improvement of this work.

I would like to thank Qassim University for all the financial support provided, and also for the scholarship that funded my Ph.D study and research.

Above all, a special thank goes out to my family. I would like to thank my late father for his inspiration, support, and encouragement. Since I was a child, he believed in me, but he did not live to see this moment. I would like to thank my mother, my wife, and my two daughters, Najd and Alanoud, for all the love they have given me. Words cannot express how grateful I am to have you in my life. Also, I would like to thank my brothers and sisters for all their support, which has helped make the life easier for me. Finally, I would like to thank my late brothers, Bader and Fasel, who I lost while studying in the United States, for all wonderful memories and joy we had together.

Dedication

Dedicated to my family. . .

Chapter 1

Introduction

1.1 Motivation

Over the last decade, the continuous growth of the Internet has resulted in a significant amount of online information, in the form of texts, images, audios, videos, and other electronic products. Meanwhile, the number of active users of online applications, the available items on business websites, and the shared content on multimedia platforms, are growing rapidly. For example, users create different accounts across different social media platforms and other business websites to share their content, to express their ideas, to find new friends, to keep themselves up-to-date with what is happening in their community and around the world, etc. Furthermore, people surf the Internet on a daily basis looking for various types of information through a wide and various range of options.

Given the huge body of information available nowadays, the process of finding the right content that is of interest to a specific user is highly challenging and beyond human ability. This is where the recommender systems come into play. Recommender systems exploit user previous information in the system to filter out irrelevant content and provide users with their personalized and desirable items based on their preferences. People utilize recommender systems for various aspects of their lives, either for their personal activities, such as finding a book to read, a place to visit, a friend to meet, a restaurant to dine in, etc., or for their

professional activities, such as looking for a job to apply for, an online course to attend, etc. When using recommender systems, people can quickly find the desired information and avoid facing a large range of options that would require lots of time and effort to look through.

In fact, the benefits of recommender systems are not limited to individual users, but also businesses and online service providers benefit from them greatly, especially when adopting successful recommendation models. First, recommender systems can assist companies to increase the user consumption of their services and products, which results in increasing their revenues. Second, using recommender systems can help companies to diversify their products by recommending various options of products to the users. Third, recommender systems reduce the problem of limited stacking (storage), where items may not be available or there is not enough space to stack more items in an actual store. Finally, businesses depend on recommender systems to maintain users satisfaction and keep them engaged in their provided services. Nowadays, recommender systems have become the cornerstone for almost all successful online service providers, and it is difficult to imagine any of the big companies being able to continue in the race without using recommender systems.

Recommender systems focus mainly on exploiting a user's available information in the system to capture historical user preferences and recommend items that meet the user's satisfaction. Almost all existing recommender systems are designed for single-domain recommendation tasks, and can face a big challenge when applied to real-world problems. Specifically, when a user has a very small number of interactions in the system, single domain recommender systems cannot generate accurate, personalized recommendations because of the sparsity problem. In other words, if there is no sufficient amount of user historical information available, the system cannot extract the user's preferences correctly, especially when using only a single source of information.

However, users interact with various domains, and thus express their opinions and share content about items in different domains. Recent studies have found that aggregating user information from multiple domains can alleviate the challenge posed by the sparsity of user interactions in one domain. Moreover, transferring user information from a dense source domain to a sparse target domain can improve recommendation accuracy in the target domain.

Due to the fact that cross-domain recommendations represent a key solution to the sparsity issue, while being relatively new, there is still an urgent need for more research in this area.

Towards this need, this dissertation concentrates on developing cross-domain recommendation models to alleviate the sparsity issue and improve recommendation accuracy in a target domain, by transferring information from a source domain. We propose three different cross-domain recommendation approaches and utilize real cross-domain datasets to investigate the effectiveness of our models.

1.2 Problem statement

Single domain recommender system models have been widely applied in e-commerce because of their popularity, availability, functionality, and effectiveness, among others. Almost all e-commerce, social media applications, and online services have adopted recommender system models to utilize their customer's data in an effective way. However, there are some situations where the available data in one domain is not enough for single domain recommender system models to function properly (e.g., for newer types of items or types of items that are not frequently consumed by users). To alleviate this problem, businesses could aggregate information from other auxiliary source domains (that focus on different but similar types of items), an approach that is generally known as cross-domain recommender systems. Let's assume that we have two domains: a *target* domain T which exhibits high sparsity in terms of user-item interactions, and a related *source* domain S which is more dense. We assume that both S and T share a set of users U, but have different types of items denoted as I_T and I_S, respectively. If a single-domain recommender system model is applied on the target domain T, this model will most likely suffer from the sparsity issue and produce inaccurate item recommendations. An effective technique is to learn user's preferences in the source domain S and utilize the shared users as a bridge to transfer the extracted information into the target domain T, and thus reduce the sparsity problem and increase the recommendation accuracy. This is the main focus of this dissertation.

1.3 Contributions and outline

In this dissertation, we propose three cross-domain recommender system approaches using deep neural networks, with the goal of reducing the problem of data sparsity in a target domain. These approaches represent three major contributions of this dissertation, as described below:

- *Cross-domain recommender systems based on neural collaborative filtering*

  First, we propose a cross-domain recommender system approach, which is based on neural collaborative filtering. We utilize users' and items' source latent vectors, learned from linear and non-linear models of the source domain, and transfer knowledge into the target domain. The knowledge transfer happens at the embedding layer by summing all items' latent vectors (specifically, latent vectors of the items the user u has interacted with in the source domain). Several source-target domain pairs extracted from a publicly available Amazon dataset (which contains items of different types, such as books, movies and music) were used to evaluate the proposed cross-domain approach. Experimental results show that the knowledge transfer from the source to the target results in an increase in the recommendation accuracy in the target domain. The approach and experimental results were recently accepted for publication in the Proceedings of the 17th International Conference on Machine Learning and Data Mining (MLDM 2021) (Alharbi and Caragea, 2021a).

- *Cross-domain self-attentive sequential recommendations*

  Second, we propose a cross-domain recommender system approach for sequential recommendation tasks. This approach takes as input an ordered sequence of items for a target user and predicts the next item that the user could potentially interact with. We utilize a state-of-the-art self-attentive sequential recommendation model to extract a user's source representational vector from the source domain. Then, the early fusion method is adopted to transfer the pre-learned user source representational vector into the target domain to improve the recommendation accuracy of the target. As for

the cross-domain recommender systems based on neural collaborative filtering, pairs of source-target sequential datasets were assembled based on the existing, publicly available Amazon dataset. Experimental results show the usefulness of the knowledge transferred from the source to the target in improving the target recommendation accuracy. A paper describing this cross-domain approach and experimental resulting showing its feasibility was recently accepted for publication in the Proceedings of the 2nd International Conference on Data Science and Applications (ICDSA 2021) (Alharbi and Caragea, 2021b).

- *Cross-Domain attentive sequential recommendations using general and current user preferences*

  Third, we propose another cross-domain recommender system approach focused on sequential recommendations, which transfers knowledge from one or more source domains into the target domain. As part of this approach, we design a general model for the source domains which is meant to extract a user's general representational vectors from the item sequences in the available sources. In addition, the self-attentive sequential recommendation model is adopted to capture the user current (dynamic) representation from the sequence of items in the target domain. Lastly, we combine the general and current target representational vectors together to predict the next target item for a user u. Experimental results on subsets extracted from the Amazon dataset show that the proposed approach significantly increases the recommendation accuracy in the target domain, as compared with all the baselines considered. A paper describing this approach and the corresponding experimental results will be submitted for publication in the Proceedings of the 15th ACM Conference on Recommender Systems (RecSys 2021) (Alharbi and Caragea, 2021c).

The rest of the dissertation is organized as follows: First, a general review of recommender systems and cross-domain recommender systems is presented in Chapter 2. Chapter 3 describes the main state-of-the-art approaches that are relevant to the proposed approaches and used as baselines in this dissertation. Then, the non-sequential cross-domain model us-

ing the learned latent vectors to transfer knowledge is described in Chapter 4. Chapter 5 presents the first sequential approach with attention mechanism to aggregate user's information from source and target domains. Finally, the second sequential approach proposed in this dissertation, which aggregating several user representations from several source domains in addition to the target domain, is described in Chapter 6. We conclude this dissertation and provide ideas for future work in Chapter 7.

Chapter 2

Background

In this chapter, we provide an overview of traditional single-domain recommender systems, as well as cross-domain recommender systems. First, we give a general introduction about recommender systems. Second, we review classifications of recommender systems based on different criteria. Lastly, cross-domain recommender systems are discussed.

2.1 Recommender systems

Due to the rapid increase of shared content on the Internet as well as millions of users surfing the Internet looking for various products, items or information, recommender systems have become an essential part in almost all e-commerce applications and social media platforms. Recommender systems aim to understand users' preferences and recommend items of interest to users, including a movie to watch, a restaurant to dine in, music to listen to, a job to apply for, etc. Despite the advantage of access to a significant amount of online data, this may lead users to frustration when they are looking for some desirable information, but they face the problem of *information overload*. Another example of information overload is where millions of users, who participate in various social media applications, share texts, pictures, or videos of their daily activities, while others can review and respond to these posts based on their interests. With the help of recommender systems, users will not be overwhelmed by

the huge number of options available on the Internet. Indeed, these enormous choices can be filtered out into a small number of personalized items, saving users' time and effort.

A general task of a recommender system is to calculate the relevance score between a user u and an item i, and then rank items accordingly. By ranking items for the user, recommender systems not only help users to avoid irrelevant items in the system, but they also help them to make their decisions faster. Furthermore, recommender systems may assist users to make decisions in some areas where they do not have enough knowledge (or tasks that required lots of time).

Recommender systems utilize user-item interaction matrices in order to generate item recommendations. An interaction matrix is a 2-D matrix, where the number of rows represents the number of users in the system, and the number of columns corresponds to the number of items, together resulting in a huge matrix that records user interactions with items, activities, or feedback. A user usually interacts with a relatively small number of items in the system, a phenomenon that leads to a very sparse matrix. The item-user interactions (input/feedback) in recommender systems can be classified into two categories: *explicit feedback* or *implicit feedback*. The explicit feedback indicates that users intentionally and explicitly provide their degree of satisfaction about an item they interacted with in the form of a rating or a review. For example, a user can give a rating of 1-to-5 or write a review about a book she just read. A popular line of work of this type is known as item recommendation based on user's ratings. However, the main drawback of using explicit feedback in item recommendation is that users generally tend not to give a rating on items. This makes it difficult to obtain enough information about the users (Hu et al., 2008; Rendle and Freudenthaler, 2014; Rendle et al., 2012).

On the contrary, implicit feedback indicates the information can be collected automatically by the system by recording each single activity the user does. Examples of user's implicit feedback include clicking an item, reading a product review, watching a movie, and so on. For example, despite the absence of explicit user ratings, Amazon tracks and employs user implicit feedback to produce recommendations for each user. In fact, Amazon uses implicit feedback, such as user purchases and user browsing history, to generate item recom-

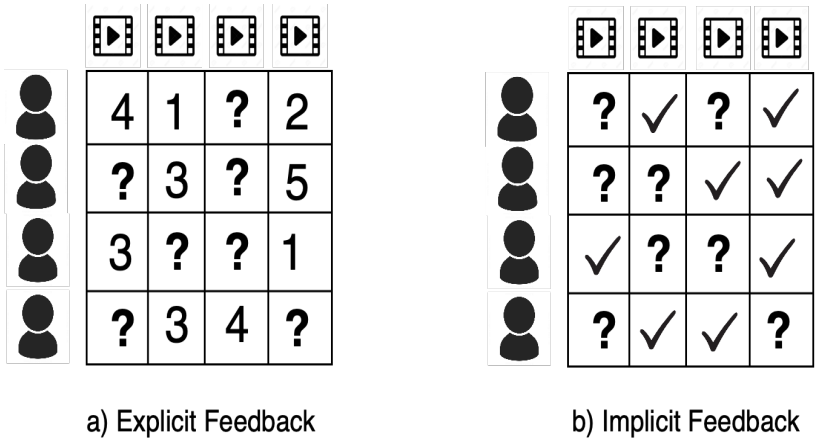


Figure 2.1: Examples of explicit and implicit user feedback employed in recommender systems

mendation under the terms *Inspired by your purchase* and *Inspired by your browsing history*, respectively. In this case of implicit feedback, the user information can be recorded in a user-item binary matrix, where the interaction of the user with an item is recorded as 1 and indicates positive feedback, while the lack of interaction is recorded as 0 and is assumed to indicate negative feedback. Alternatively, one can record the number of interactions with an item, which allows to rank items according to user preference based on the count. In recent years, the recommender systems' literature has been shifting toward utilizing user implicit feedback, as it is easier to obtain at a large scale (Bayer et al., 2017; He et al., 2016b). The difference between user explicit and implicit feedback is illustrated with examples in Figure 2.1.

2.2 Types of recommender systems based on information used

Due to the ability of recommender systems to tackle the problem of information overload, there have been a considerable number of research studies focused on designing and developing recommender system models. As item recommendation is based on filtering out items for users, the literature generally classifies recommender system models into three categories:

- **Content-based (CB):** The content-based filtering approach relies heavily on item content/profiles represented as item features to associate new items with a specific user based on similar items that the user has interacted with in the past. More specifically, each item in the dataset is represented with a set of features that describe the item's characteristics. For a movie dataset, as an example, movies can be represented with a variety of features, such as genre, director, actors, runtime, etc. The recommender system first calculates item-item similarities based on item attributes, and then it can recommend a list of items that are similar to the ones a user has interacted with. For example, the recommender systems could recommend a new movie that is similar in some respects to the ones the user previously watch.

- **Collaborative filtering (CF):** Unlike the content-based approach, collaborative filtering relies only on information about previous user preferences on items to generate recommendations. Collaborative filtering utilizes user's historical behaviors stored in the system to distinguish similar users and recommends some items that similar users have interacted with or highly preferred. Most available datasets only contain user-item interactions and they are suitable for the collaborative filtering approach, as the content-based approach requires additional features about items and/or users.

- **Hybrid-based:** The hybrid approaches use a combination of content-based and collaborative filtering, benefiting from both types of information that those individual approaches use. Hybrid recommender system models have been proven to result in better recommendation performance when compared to only content-based or collaborative filtering.

In the following subsections, we will discuss the advantages and drawbacks of these approaches, as well as review some previous relevant works.

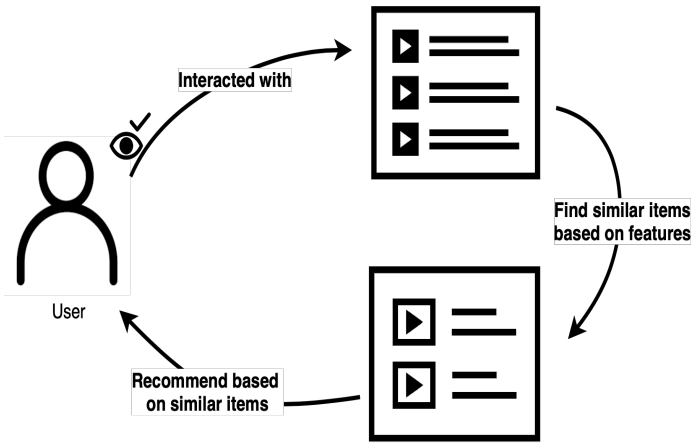


Figure 2.2: Overview of content-based recommender systems

2.2.1 Content-based methods (CB)

Content-based recommendation utilizes item content (item-related features) to suggest similar items to the user based on the past items the user has interacted with (Lops et al., 2011; Pazzani and Billsus, 2007). Specifically, using attributes of the items that the user rated or interacted with in the past, the system recommends new items that share similar attributes (Lops et al., 2011). Figure 2.2 provides a visual description of how the content-based approach works.

One advantage of the content-based recommender systems is that it does not require learning about other user's preferences to perform recommendations (as the collaborative filtering does). Instead, it only learns about a specific user and his/her previously preferred items. As a consequence, this approach does not need to deal with the spareness in terms of information about other users in the system. Another benefit of using the content-based method is that the system can provide a justification (in terms of previously preferred items) about the recommended items and how these items are relevant to that specific user. For example, the model can list the common item features that were considered when making the recommendation.

The main drawback of the content-based approach is that this method limits the scope of the recommendation process by only using the information about items available in the user's profile. In other words, item recommendation is generated based only and fully on user's

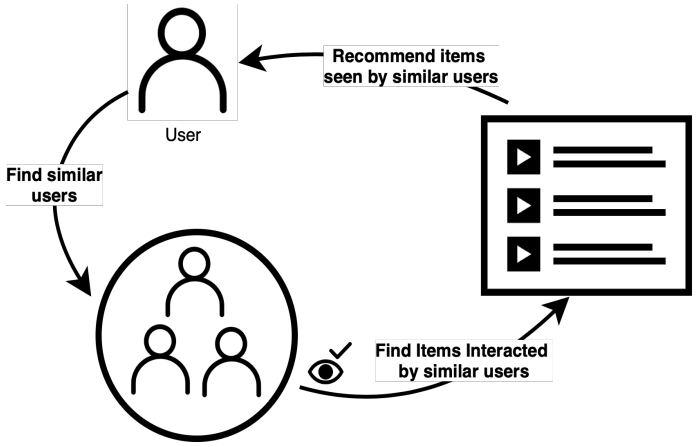


Figure 2.3: Overview of collaborative filtering approaches to recommender systems

past preferred items, leading to a problem known as over-specification. In some situations, too similar items could be recommended in a way that is not satisfactory for the users. For example, a user may receive two news articles from different websites or two YouTube videos from different channels that contain very similar information about an event. More importantly, the user and item profiles may not be well-developed, and constructing these profiles requires significant time and effort. Also, the recommendation performance can be hindered if the amount of item content is not sufficient (Shardanand and Maes, 1995). Lastly, without enough information available about the users (e.g., new users added to the system who have not interacted with many items), content-based models cannot provide accurate recommendations (Lam et al., 2008).

2.2.2 Collaborative filtering (CF)

Collaborative filtering is one of the most popular types of recommender system approaches. It concentrates on observing and analyzing user-item previous interactions in the system to capture the similarity between users, extract users' preferences, and then recommend items based on the interests of other similar users in the system (Ricci et al., 2011). An overview of collaborative filtering approaches is shown in Figure 2.3. The main advantage of collaborative filtering is that it relies on previous user-item interactions, that can be automatically collected, especially in the case of implicit feedback. Hence, it does not require additional

information about item features and characteristics, as in content-based methods. However, as mentioned above, a major limitation of this approach is the sparsity problem. Collaborative filtering models can be further categorized into two categories (Su and Khoshgoftaar, 2009):

- **Memory-based collaborative filtering:** This approach calculates the similarity between users or items in order to make recommendations. It is also known as neighborhood-based recommendation (Desrosiers and Karypis, 2011). There have been many works based on either user-user collaborative filtering or item-item collaborative filtering. The models based on user-user collaborative filtering assume that users who share similar interests on previous items they have interacted with are more likely to share similar interests in the future (Resnick et al., 1994). Despite its effectiveness, the system is required to calculate all user-user pairs, which is computationally expensive, especially in large user-active systems where more new users are expected to join or to be added all the time. Furthermore, users naturally have dynamic preferences that change over time and that may require the system to re-calculate the user-user similarly, when receiving more user-item interactions, to provide more accurate recommendations. With static features, item-item collaborative filtering models seem to have a simpler computation when calculating the item similarity (Sarwar et al., 2001). This type of system assumes that if a user expresses an interest toward a specific item, that user will also express interests toward other similar items. Memory-based collaborative filtering generally uses two popular similarity metrics: the *Pearson correlation* or the *cosine similarity* to calculate user-user or item-item similarity (Deshpande and Karypis, 2004).

- **Model-based collaborative filtering:** Compared to memory-based collaborative filtering, the model-based collaborative filtering is a more sophisticated approach that builds a model to predict the relevance score between a user u and an item i, indicating how likely it is to recommend item i to user u (Koren and Bell, 2015). The prediction model can be either a rating prediction or a ranking prediction. The rating prediction

models are built to output the expected rating of a user for an item (Bennett et al., 2007), using mostly explicit feedback. On the other hand, the ranking prediction models are designed to provide a user with top-N item recommendations (Aggarwal et al., 2016), where the N items are ranked based on the user's preferences, using mostly implicit feedback. Using machine learning, the models try to learn user's pattern over the available data, representing users and items by latent vectors (Wang et al., 2015). This is why this type of model is sometimes called latent-based model. The most popular method in the latent-based model category is Matrix Factorization (MF), which represents users and items using low dimensional vectors. Specifically, the matrix factorization approach decomposes the user-item matrix into two smaller matrices, a user matrix and an item matrix (Koren and Bell, 2015). The user matrix contains user latent vectors representing user factors, while the item matrix contains items vectors representing items features. In a movie dataset, as an example, user factors can represent latent information about user's age, gender, etc. On the other hand, item factors represent latent information about item's genre, actors, directors, run-time and other features relevant to a movie.

2.2.3 Hybrid methods

In order to avoid the limitations faced by either content-based or collaborative filtering approaches and build more robust models, hybrid recommender systems combine both content-based and collaborative filtering methods together (Ghazanfar and Prugel-Bennett, 2010). This results in models with better performance and more diverse item recommendations. In fact, many large companies may not be able to come out with recommendations that are satisfactory to their customers without diversifying their recommendation models. There are various existing forms of hybrid recommender system applications, where the combined models can operate in parallel, in sequence, or even switching between the parallel and sequence modes according to specific situations. For example, some companies balance the outputs of the two types of models when making the final recommendations, while others

use the outputs of one model as the input to another (e.g., cascading) (Lampropoulos et al., 2012). Coupling collaborative filtering with other types of recommendation models is the most popular way to achieve a hybrid approach that can cope with the sparseness of user and/or item information in the system.

2.3 Recommender systems based on deep learning

In recent years, Deep Learning (DL) approaches have become state-of-the-art in many applications areas, including (but not limited to) Computer Vision (Yosinski et al., 2014), Speech Recognition (Hinton et al., 2012), Natural Language Processing (Yang et al., 2017), Question Answering (Kumar et al., 2016; Xiong et al., 2016). As a consequence, many researchers have devoted a lot of efforts in developing recommender system models using deep learning techniques. Unlike traditional models, that learn based on features that are manually designed, deep learning extracts representations at different levels of abstraction for users and items based on the available raw training data (Wang et al., 2015). A main advantage of deep learning recommendation models is the ability to capture non-linear relations between users and items (He et al., 2017b; Wang et al., 2015).

The earliest recommender systems based on deep learning utilize Restricted Boltzmann Machines (RBM) (Salakhutdinov et al., 2007), which model user-item explicit ratings by applying a visible layer and a hidden layer. Several other works utilize auto-encoders (AE) (Sedhain et al., 2015; Wu et al., 2016), which are feed-forward neural networks used to learn hidden representations and reduce dimensionality. Auto-encoders compress a large size input through a bottleneck in the architecture constricting a smaller hidden layer, which learns a latent representation of the input. Then, from the hidden layer, the model learns to reconstruct the original input. Later models used the learned latent vectors from auto-encoders as inputs to both traditional and other deep learning models. Another feed-forward neural network that utilizes a non-linear activation function in each layer is the traditional Multi-Layer Perceptron (MLP) network. Multi-layer perceptron networks have been used to replace the inner product of matrix factorization with the goal of allowing the model to capture com-

plex features from user-item interactions, as done in Neural Network Matrix Factorization (NNMF) and Wide & Deep models (Cheng et al., 2016; Dziugaite and Roy, 2015). Neural Collaborative Filtering (NCF) concatenates the output of a multi-layer perceptron network with the learned latent vectors from matrix factorization to produce a better final representation, which benefits from both linear and non-linear functions (He et al., 2017b). To endow the model with more flexibility, the neural collaborative filtering model generates different user and item embedding layers for multi-layer perceptron network and for matrix factorization, instead of using shared embeddings. Compared with pure matrix factorization or multi-layer perceptron networks, the neural collaborative filtering model shows superiority in terms of recommendation accuracy.

To emphasize the importance of order (and time) when making decisions, Recurrent Neural Networks (RNN) are designed to capture temporal features (Donkers et al., 2017; Hidasi and Karatzoglou, 2018). RNNs extract the user's preferences at each time interval and used hidden vectors to handle the captured information throughout the item sequence. A session-based recommendation model adopted the Gated Recurrent Unit (GRU) model to distill information through long sequences (Hidasi et al., 2015). To model the users' short preferences more effectively, Convolution Neural Networks (CNN) were used to learn user-item interactions that capture spatial representations, utilizing $n \times n$ sized filters (Tang and Wang, 2018). Despite all their success, these models face difficulty in terms of propagating user useful information when considering very long sequences, without losing meaningful information.

Recently, the attention mechanism has been proposed to tackle the vanishing gradient problem of RNNs (Vaswani et al., 2017). By weighing the importance of all items in the sequence, attention-based models aggregate the user's representational vector based on the most relevant (important) items. Unlike RNN and CNN networks, that compute hidden states sequentially, attention-based models can compute hidden states in parallel, resulting in high computation efficiency, in addition to a better recommendation accuracy. One model in this category is the self-attention sequential recommendation (Kang and McAuley, 2018). When outputting a hidden vector of an item $m+1$ at a specific time interval, this aggregates

only the link associated with previous m items.

As all the above recommender system models are specialized to a single domain, they suffer from the problem of data sparsity. Consequently, they cannot generate accurate recommendations for users with very few interactions in the system.

2.4 Classification of recommender systems based on the nature of the available user-item interactions

Recommender system models differ in the way they structure their inputs (user interactions). Based on observing user-item interactions, the recommender system models can be categorized into two main classes: *non-sequential models* and *sequential models*. In the next subsections, we will define and discuss the advantages of each class.

2.4.1 Non-sequential models

Non-sequential models consider user-item interaction as sets, and form a user-item pair for each positive interaction. For example, matrix factorization and multi-layer perceptron networks obtain general representations of users and items using inner products or hidden layers, respectively. Sequential models are efficient in terms of capturing the relevance score between users and items by using a user-item pair at each iteration to predict the output score. However, they assume all user-item pairs to be equally important, although this assumption is not correct. For example, if a user u has watched a movie i a month ago and also watched another movie j last night, the non-sequential systems treat both interactions in the same manner, given them equal contributions when making the next item recommendation, although it would make more sense to give a higher weight to the more recent movie.

2.4.2 Sequential models

Sequential models observe user-item interactions in a temporal sequence, reflecting the real-world situation in which a user experiences items in a sequence. RNNs, CNNs, and attention-based sequential models are good examples of learning user's preferences over time. These models are designed to cope with the dynamic changes of user's preferences over time. For example, people may change their interest toward an item based on an occasion, a season, or even a time during the day. Sequential models consider an additional dimension (specifically, time or order) to track the order of the user's behavior, while capturing the user's long and short preferences. At each time, the user's preferences are propagated from previous interactions to accurately and precisely predict the next item that a user is most likely to engage with.

2.5 Cross-domain recommender systems

As mentioned before, recommender systems have been increasingly important in almost all e-commerce websites and social media applications. Thus, all these companies have invested significantly in the area of recommender systems in order to keep their costumers satisfied by providing them with more precise recommendations, and consequently increase their revenues. Most existing recommender system models are designed to exploit a single domain's information to recommend items in that domain to users. As mentioned above, single-domain recommendations suffer greatly from the sparseness of user-item interactions. An effective solution can be obtained by transferring knowledge from other source domains, where there is more information about the users, as that can help reduce the sparsity issue and increase recommendation accuracy in the target domain. This solution is generally known under the name of cross-domain recommender systems. Such systems have been recently proposed to tackle the sparsity issue and cold-start users (Khan et al., 2017).

There have been several works focused on the classification of cross-domain systems in terms of the distinct domains used for transferring knowledge (Cantador et al., 2015; Khan

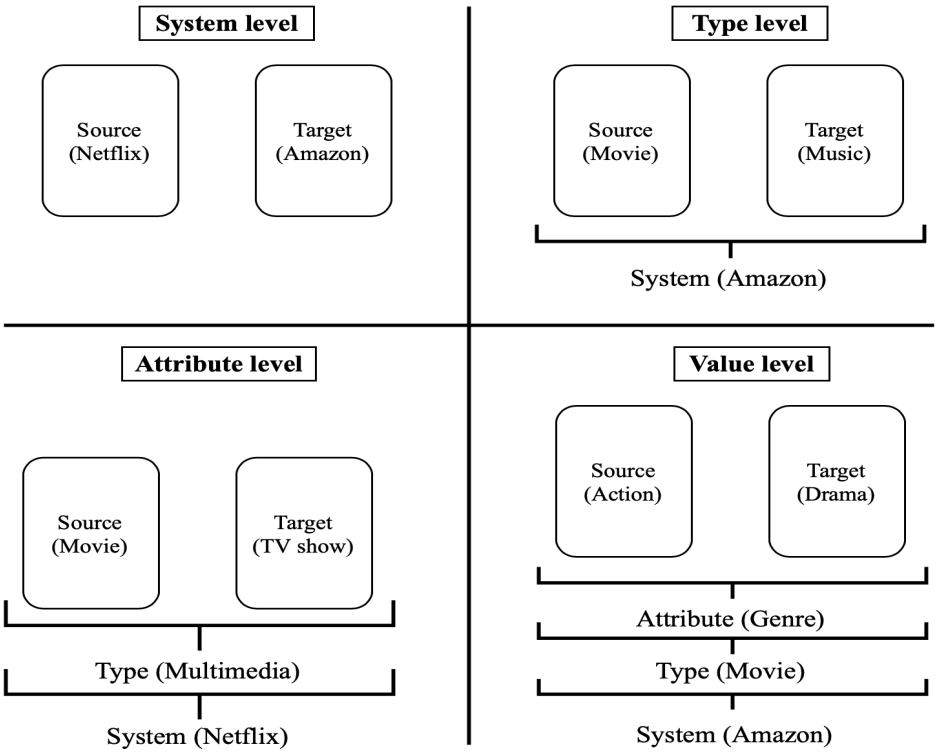


Figure 2.4: Source-target domain classification based on the level at which the domains are considered. Examples for each source-target category are also shown. [Figure adapted from (Cantador et al., 2015)].

et al., 2017). A major classification by Cantador et al. (2015) considers four categories (based on different levels of granularity) to distinguish any two separate source-target domains. A high-level overview of the four categories is shown in Figure 2.4. More details about each category are provided in what follows.

- *System level:* If items belong to separate systems (e.g., two different businesses), they are considered to form two different system-level source-target domains. For example, Movies on Amazon and Movies on Netflix can be considered as distinct system-level domains.

- *Item level:* If items belong to the same system (e.g., same business), but have different types, they are considered to form two different type-level source-target domains. For instance, Books and Movies on Amazon correspond to two type-level domains.

- *Attribute level:* If items belong to the same system and have the same type, but they

do not share all the attributes, they are considered to form two different attribute-level source-target domains. For example, Movies and TV shows can fit in this category as they represent items in the same domain, of the same type, that share some attributes (e.g., title), while other attributes may be different.

- *Value level:* If items belong to a unique system, have the same type and the same set of attributes, but the values of the attributes differ, they are considered to form two different value-level source-target domains. For example, different Movies on Amazon may have different values of the attribute "genre".

In order to effectively transfer knowledge across different domains, a cross-domain recommender system should build a bridge between the source and target domains to enable the information flow from the source to the target. The bridge could be achieved through the users and/or items that the two domains might share. Cantador et al. (2015) classify source-target domains used in cross-domain recommender systems based on several user-item overlap scenarios, as described in what follows. A graphical representation of the overlaps is also shown in Figure 2.5.

- *No-overlap:* In this scenario, there is no user/item overlap between the source and target domains. Thus, the users/items can't be used as a bridge in this scenario. However, cross-domain recommender systems can analyze the rating patterns of the users and items from the two domains to capture some similarity and transfer some knowledge.

- *User-overlap:* In this scenario, there is a set of users who have interacted with different items from the two different domains. Cross-domain models can use the shared users as a bridge to transfer knowledge and alleviate the sparsity issue of the target domain.

- *Item-overlap:* In this scenario, a set of items is shared between the two different domains, although they are not sharing any users. Similar to the previous scenario, the items that overlap in the two domains can be utilized as a bridge for transferring knowledge from the source domain to the target domain.

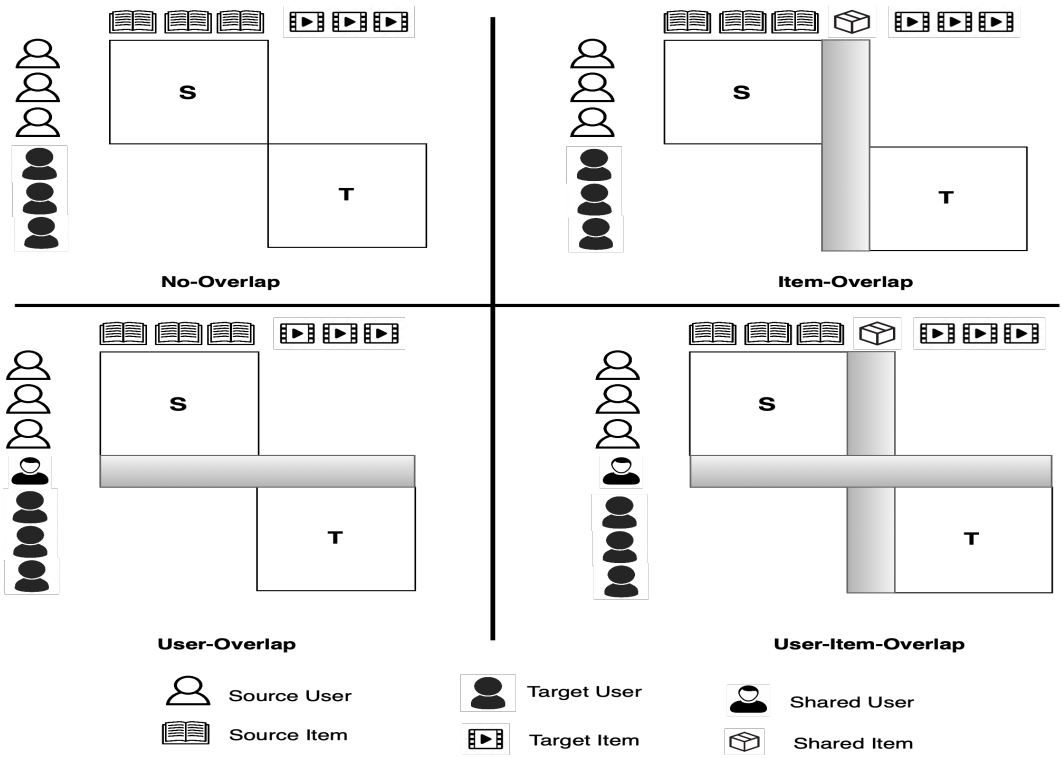


Figure 2.5: Overlap scenarios between source and target domains in a cross-domain recommender systems [Figure adapted from (Cantador et al., 2015)].

- *User-item overlap:* In this scenario, there are both common users and common items shared between the source and target domains.

The scenarios where users or items overlap between source-target (or even multiple) domains are more likely to assist in eliminating the data sparsity, as well as generating more precise recommendations to the shared users, in particular.

Given the description of the source-target domains based on different levels of granularity, and their relationship in terms of partially or fully user/item overlaps, next, we describe the major techniques used for allowing information to flow between different domains. Specifically, there are two major methods (Cantador et al., 2015; Khan et al., 2017):

- *Knowledge aggregation:* This technique is focused on identifying information that can be aggregated between the two domains. For example, the aggregated information can be the items that a shared user has interacted with in both domains. This aggregated knowledge is used to predict and recommend items.

- *Knowledge transfer:* This technique is focused on connecting the source domain with the target domain using a bridge for the information to flow. This bridge can be the overlapped users. Thus, the transferred information from the source domain is utilized to increase recommendation accuracy in the target domain.

2.5.1 Summary

In this chapter, we first provided an overview of single-domain recommender systems. We described how the user-item interactions could be collected in the form of either implicit feedback or explicit feedback. Also, we discussed different classifications of recommender systems in terms of the information they use, the type of interactions (temporal or non-temporal), etc. We mentioned that traditional models usually fall into one of these categories: content-based, collaborative filtering, or hybrid-based. Non-sequential recommender system models observe each user-item interaction individually, while sequential recommender system models consider the order of the item sequence. Finally, we described cross-domain recommender systems, which represent an effective solution to transfer knowledge across domains and help alleviate the sparsity problem. We also defined different scenarios encountered in cross-domain recommender systems.

Chapter 3

Related work

In this chapter, we present the main related works on single-domain and cross-domain recommender systems. Further, we provide detailed information about each model and its components.

3.1 Matrix factorization (MF)

Matrix Factorization (MF) is one of the most commonly used CF-based models for recommender systems. In fact, the use of MF is not limited to recommender systems but also broadly used in computer vision (Yosinski et al., 2014) and Natural Language Processing (Yang et al., 2017). Generally, MF embeds users and items into low dimensional vectors, and performs the dot product to project user-item ratings into the latent space. Specifically, MF factorizes the user-item interaction matrix $A = \mathbb{R}^{|U| \times |I|}$ into two smaller embedding matrices with dimension d, simply called user matrix $P = \mathbb{R}^{|U| \times d}$ containing user's learned latent factors and item matrix $Q = \mathbb{R}^{|I| \times d}$ containing item's learned latent factors. Thus, each $p_u \in P$ is a latent vector of factors representing the preference of user u. Similarly, each $q_i \in Q$ is a latent vector of factors representing the features of item i. The dot product (or inner product) of these two vectors p_u and q_i provides the overall interaction between user u and item i. Thus, the general formulation of MF (the projection or mapping function of

dot product) is as follow:

$$d_{ui} = f(p_u, q_i) \tag{3.1}$$

$$f(p_u, q_i) = p_u^T q_i = \sum_{k=1}^{d} p_{uk} q_{ik} \tag{3.2}$$

where d_{ui} indicates the relevance score of user u on item i. In other words, items are ranked for a user u if their relevance scores d_{ui} are higher.

Matrix Factorization can also include some biases for users and items in terms of ratings, in which case, the dot product is described by Gogna and Majumdar (2015).

$$d_{ui} = f(p_u, q_i) + b_u + b_i + b \tag{3.3}$$

where b_u represents user bias, b_i represents item bias, and b is the average ratings. In the ranking item prediction using implicit feedback, b_u and b are removed because this task is based on item ranking. Despite its variants, MF is based on a simple linear function that cannot capture the complex features of users and items.

3.2 Generalized matrix factorization (GMF)

As mentioned above, the obtained embedding vectors of users and items using MF lack the ability to capture complex interactions between users and items. Therefore, He et al. (2017b) introduce Generalized Matrix Factorization (GMF) that utilizes a nonlinear activation function and latent vector h to upgrade MF. The output of GMF (the relevance score) can be computed by the following equations:

$$d_{ui}^{GMF} = \sigma(h^T(p_u \odot q_i)) \tag{3.4}$$

where $\odot$ denotes the element-wise product and σ denotes a non-linear activation function, e.g., Rectified Linear Unit (ReLU) used at the output layer. Also, this formulation uses

latent vector h that is learned from the data. Thus, GMF endows MF with the ability to learn user-item interaction using a nonlinear function. The right side of Figure 3.1 shows the GMF model.

It is worth mentioning that GMF can be degenerated into the simple MF if σ is set as the identity function and h is constrained to be a vector of 1s.

3.3 Multi-layer perceptron (MLP)

To overcome the limitation of MF, previous work (Cheng et al., 2016) replaces MF with neural network layers, resulting in Multi-Layer Perceptrons (MLP) networks. An MLP is capable of learning more complex and hidden properties of user-item interactions. The main idea of MLP is to concatenate the user's latent vector and item latent vector and then feed this concatenated vector $E_{ui} = [p_u, q_i]$ through a series of fully connected layers, such that the output of the preceding layer is used as the input of the following layer. The MLP layers can be defined by the following equations:

$$d_{ui}^{MLP} = f(E_{ui}|P, Q, \theta_f) \tag{3.5}$$

$$f(E_{ui}|P, Q, \theta_f) = \phi_o(\phi_L(...\phi_2(\phi_1(E_{ui}))...)) \tag{3.6}$$

where ϕ_o and ϕ_L are the functions to compute the output layer and the L-hidden layers of MLP, respectively. Also, θ_f denotes the model parameters. The common configuration of MLP network is a tower pattern, in which the size of a current layer is half the size of the prior layer. The left side of Figure 3.1 shows the configuration and the structure of an MLP. Also, to prevent the MLP model from over-fitting, common regularization methods (e.g., dropout) are usually applied.

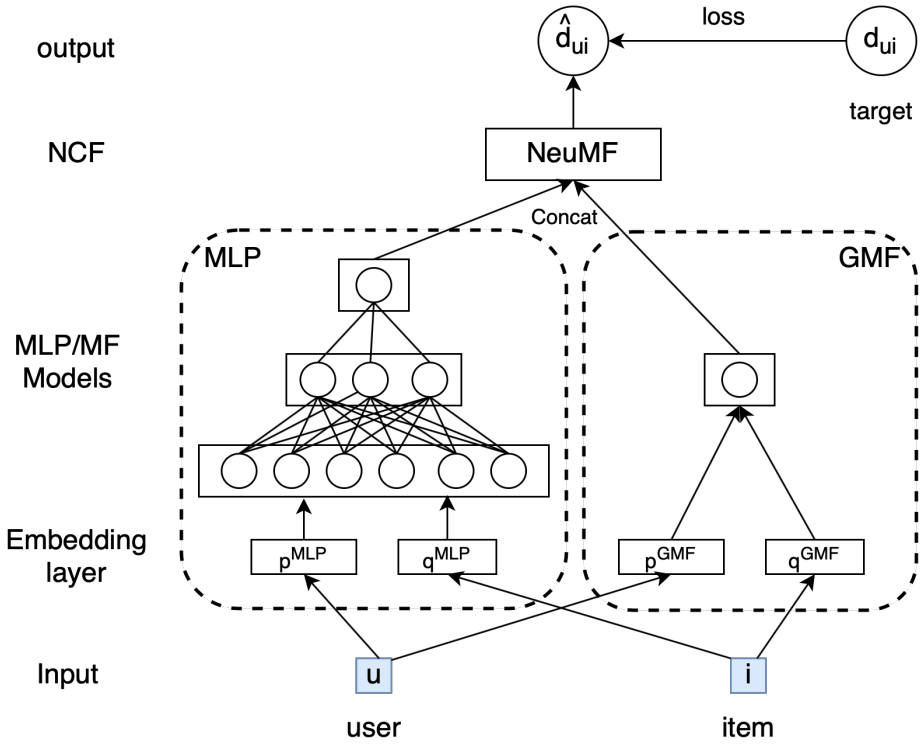


Figure 3.1: Neural collaborative filtering (NCF) [Figure adapted from (Cheng et al., 2016; He et al., 2017b)].

3.4 Neural collaborative filtering (NCF)

Neural Collaborative Filtering (NCF) was introduced by He et al. (2017b) to overcome the MF's expressive limitation, as MF uses a simple and fixed inner product function to project users and items into the latent space. He et al. (2017b) amends and upgrades the simple MF into GMF as explained in Section 3.2, by leveraging non-linear activation functions. NCF concatenates the output of the MLP network with the inner product of GMF. The complete process of NCF can be seen in Figure 3.1. MLP and GMF learn separate embedding vectors that provide the model with more flexibility. NCF can be defined by the following equations:

$$o^{GMF} = (p_u^{GMF} \odot q_i^{GMF}) \tag{3.7}$$

$$o^{MLP} = \phi_L \left(W_L^T(\phi_{L-1} \left(...\phi_2 \left(W_2^T \begin{bmatrix} p_u^{MLP} \\ q_i^{MLP} \end{bmatrix} + b_2 \right) ... \right) + b_L) \right) \tag{3.8}$$

$$d_{ui} = \sigma\left(h^T \begin{bmatrix} o^{GMF} \\ o^{MLP} \end{bmatrix}\right) \tag{3.9}$$

where p_u^{GMF} and q_i^{GMF} are the embedding vectors for user u and item i for GMF, respectively. Similarly, for MLP, there are p_u^{MLP} and q_i^{MLP} embeddings for user and item, respectively. The activation function σ is the commonly used non-linear function ReLU. During the training, the model generates four random negative samples for each user-item positive interaction. Negative sampling is an effective, commonly used technique that assists in model training.

Since NCF is based on binary classification using implicit feedback, it uses the binary cross-entropy loss as its objective function, defined by the following equation:

$$L_f = -\sum_{(u,i)\in D^+\cup D^-} d_{ui} \log \hat{d}_{ui} + (1 - d_{ui}) \log(1 - \hat{d}_{ui}) \tag{3.10}$$

where D^+ denotes the positive user-item interactions and D^- denotes the sampled negative user-item interactions.

3.5 Collaborative cross-network for cross-domain recommendation (CoNet)

Hu et al. (2018) introduced a dual transfer knowledge across domains by utilizing two MLP networks that model each domain, respectively, and allowing information to flow from one domain to another by using cross-stitch connection networks Misra et al. (2016). Let us assume we have two domains, S and T. CoNet transfers information from domain S to improve recommendation accuracy in the domain T. At the same time, it also allows information to flow from domain T to increase recommendation accuracy in domain S. Figure 3.2 describes the structure of the CoNet model where dotted (red) arrows between the two MLP models indicate that information is transferred in the (middle) of the neural network layers. Let m denote the representation of the L-th hidden layer, and $\hat{m}$ denote the representation of the L+1-th hidden layer. The output of the cross-connection units for the two domains are

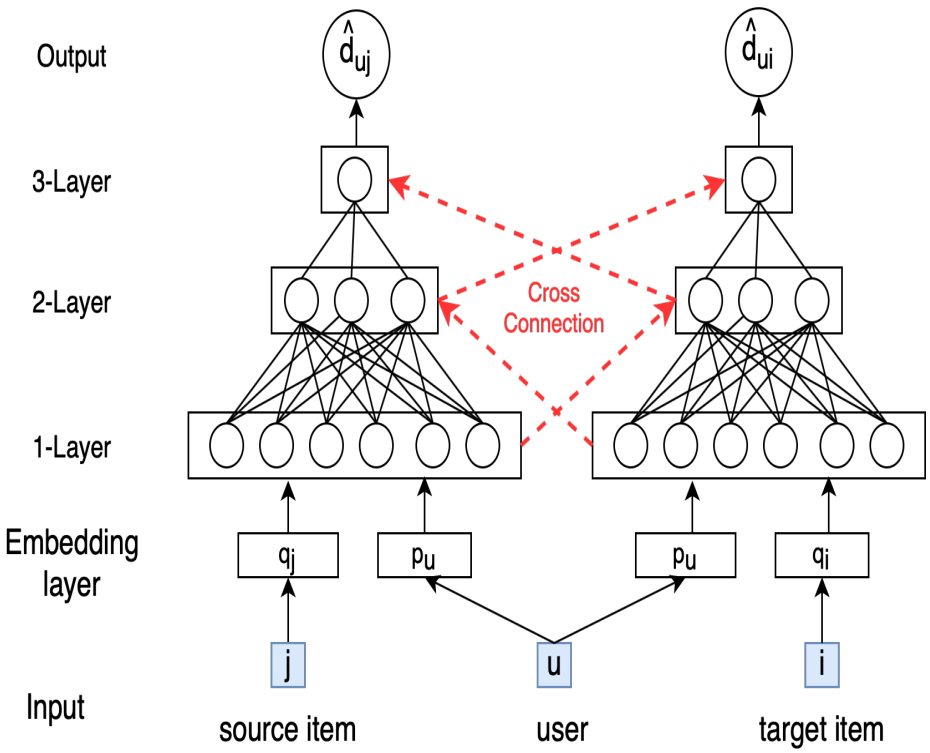


Figure 3.2: Collaborative cross network for cross-domain recommendation (CoNet) [Figure adapted from (Cheng et al., 2016; Hu et al., 2018)].

calculated by the following equations:

$$\hat{m}_S = W_S m_S + H m_T \tag{3.11}$$

$$\hat{m}_T = W_T m_T + H m_S \tag{3.12}$$

where W is the weight matrix, and H is used to control the information flow from the other domain. Thus, each following layer in the MLP network receives two pieces of information, one is coming from the previous layer in the same domain, and the other one is coming from the previous layer in the other domain. We can rewrite the cross-connection network as follow, using σ as the activation function (ReLU):

$$m_S^{L+1} = \sigma(W_S m_S^L + H m_T^L) \tag{3.13}$$

$$m_T^{L+1} = \sigma(W_T m_T^L + H m_S^L) \tag{3.14}$$

CoNet uses the cross-entropy loss function for binary classification shown in Equation 3.10 to instantiate the losses of domains S and T, as L_S and L_T, respectively. Finally, the joint loss of CoNet is defined as follows:

$$L(\theta) = L_S(\theta_S) + L_T(\theta_T) \tag{3.15}$$

where is θ denotes the model parameters of domain S and domain T, respectively.

3.6 Session-based recommendations with recurrent neural networks (GRU4Rec)

Hidasi et al. (2015) introduced GRU4Rec for session-based recommendation using Recurrent Neural Networks (RNN) as an attempt to solve the vanishing gradient of the vanilla RNN model. Previous RNN models face a problem with propagating useful information throughout a long sequence to predict the future itema. GRU4Rec utilizes a Gated Recurrent Unit (GRU) to distill the information throughout the user interaction sequence without losing the user's long-term preferences. For each item i in the sequence, GRU4Rec outputs the probability distribution of the next item $i + 1$, by using the embedding of i and the propagated (aggregated) information in the previous hidden state with a control parameter over this information. The following equation is used to update hidden state h for a simple RNN model:

$$h_k = g(W i_k + U h_{k-1}) \tag{3.16}$$

where g is the logistic sigmoid function and i_k is the input at time k. RNN takes the input i_k and the previous hidden state h_{k-1} to predict the current hidden state h_k. However, GRU uses a gate to decide when and by how much information to update the hidden state using

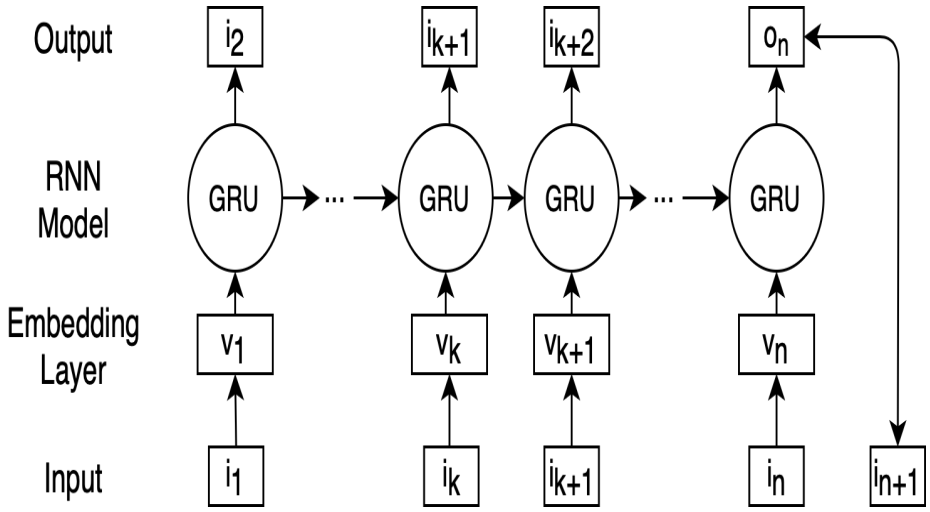


Figure 3.3: Recurrent neural network model for session-based recommendation using gated recurrent unit (GRU4Rec) [Figure adapted from (Hidasi et al., 2015; Sun et al., 2019)].

an updated gate z_k and a candidate activation function $\hat{h}_k$. Thus GRU4Rec is calculated by the following equations:

$$h_k = (1 - z_k)h_{k-1} + z_k\hat{h}_k \tag{3.17}$$

$$z_k = \sigma(W_z i_k + U_z h_{k-1}) \tag{3.18}$$

$$\hat{h}_k = \tanh(W i_k + U(r_t \odot h_{k-1})) \tag{3.19}$$

$$r_k = \sigma(W_r i_k + U_r h_{k-1}) \tag{3.20}$$

where r_k is the reset gate and it is used to reset specific hidden state as sessions are assumed to be independent.

3.7 Personalized top-n sequential recommendation via convolutional sequence embedding (Caser)

Personalized top-n sequential recommendation via convolutional sequence embedding (Caser), introduced by Tang and Wang (2018), models the user's recent items as an image to learn user short-term preferences using a convolutional filter of size $n \times n$ and then predicts the next item(s). Caser is a top-n sequential recommendation model that utilizes Convolutional Neural Networks (CNN) to learn user's sequential features. It applies a sliding window of size $L \times T$ over the user sequence, where L denotes the L previous items (e.g., 5 items in their case) taken as an input, to regard the next T target items (e.g., 3 items in their case). In other words, during the training, Caser takes an input of 5 items to predict the next 3 items. Therefore, the embedding matrix is a matrix of size $L \times d$, denoted as $E \subset \mathbb{R}^{L \times d}$ where d is the dimension of the embedding.

Caser uses a horizontal convolutional layer with n horizontal filters over the embedding matrix. The horizontal filters are defined as, $H^k \in \mathbb{R}^{h \times d}, 1 \leq K \leq n$, where $h \in \{1, 2, ..., L\}$ is the height of a filter. The interaction of the i-th convolution value can be calculated by the following equation:

$$c_i^k = g_c(E_i : i + h - 1 \odot H^k) \tag{3.21}$$

where $\odot$ is the inner product, while g_c is the activation function. After applying the horizontal convolutions layer (H^k) on the matrix, the final result of the convolution is as follows:

$$c^k = [c_1^k c_2^k \dots c_{L-h+1}^k] \tag{3.22}$$

Then, a max-pooling is adopted to obtain the maximum value among all values that were produced by each filter. The most significant feature is the one with the maximum value. Thus, the output of this layer, for all n filters, is a vector containing the maximum value of each c^k as described by the following equation:

$$o^H = max(c^1), max(c^2), \ldots, max(c^n) \tag{3.23}$$

Caser also uses a vertical convolutional layer with n vertical filters, $V^k \in \mathbb{R}^{L\times 1}, 1 \leq k \leq n$. These filters slide over the embedding matrix from left to right for d times. The final result is a vector of size d as shown below:

$$\dot{c}^k = [\dot{c}_1^k \dot{c}_2^k \ldots \dot{c}_d^k] \tag{3.24}$$

The results of the inner product are the weighted sum over each L rows:

$$\dot{c}^k = \sum_{l=1}^{L} V_l^k \cdot E_l \tag{3.25}$$

where E_l denotes the l-th row of the embedding matrix. These n filters are to capture the weighted sum over all users. Then, the final output of the vertical layer is a vector containing n weighted sum corresponding to each filter as follows:

$$o^V = [\dot{c}^1 \dot{c}^2 \ldots \dot{c}^n] \tag{3.26}$$

Note that in the case of the vertical layer, they do not apply max pooling because it is important to keep the aggregation of each latent dimension. After that, the extracted horizontal and vertical vectors are concatenated and then fed through a fully connected layer to output the final representation of the user sequence y using the following equation:

$$y = a(W \begin{bmatrix} o^H \\ o^V \end{bmatrix} + b) \tag{3.27}$$

where a is the activation function, and b is the bias vector. Finally, it concatenates y with the learned latent vector of the user P_u, and feeds this concatenated vector through another fully-connected layer to output the probability of the interaction between user u and item i as in the following equations:

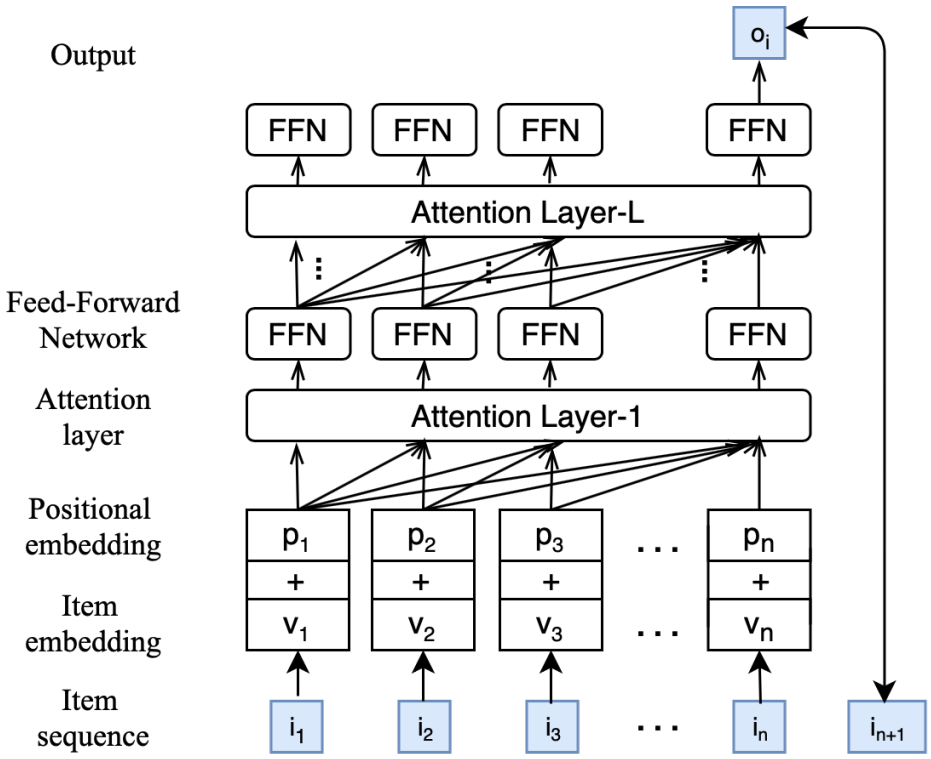


Figure 3.4: Self-attentive sequential recommendation (SASRec) [Figure adapted from (Kang and McAuley, 2018)].

$$d_i^{u,t} = W \begin{bmatrix} y \\ P_u \end{bmatrix} + b \tag{3.28}$$

where $d_i^{u,t}$ denotes how likely it is that user u will interact with item i.

3.8 Self-attentive sequential recommendation (SASRec)

The attention mechanism was designed to deal with the vanishing gradient problem of RNN models. One of the first attention models was introduced by Vaswani et al. (2017) that applied importance weights on the items in the sequence to capture a better representational vector representation of the sequence. Also, the attention mechanism enables the computation to be done in parallel in order to calculate the hidden state of each item with high efficiency, unlike RNN models that perform their computations in a sequence that degrades

the performance of the model, especially with very long sequences. The Self-Attentive Sequential Recommendation (SASRec) model, introduced by Kang and McAuley (2018), is a special case of the attention mechanism that constrains the output calculation of an item i to aggregate only the links coming from the previous $i-1$ items. The process of the SASRec model is shown in Figure 3.4. SASRec adopts a series of block attention layers, where each block attention layer consists of the actual attention layer followed by a feed-forward network. At each time interval, SASRec tries to output the next item the user will engage with. Kang and McAuley (2018) show that SASRec exhibits up to 10x speed improvements when compared to previous CNN/RNN-based models. We will further provide a thorough detail of the SASRec model in Section 5.3.2.

3.9 Summary

In this chapter, we presented the most related work to ours for both single and cross-domain recommender systems. These related models are designed based on various methods, including Matrix Factorization (MF), Multi-Layer Perceptrons (MLP), Recurrent Neural Network (RNN), Convolution Neural Networks (CNN), and the Attention mechanism, and are used as baselines for our proposed cross-domain recommender system models. We provided detailed information on how each one of these models works in terms of predicting the next item i for the user u. Our baselines include other simpler models, such as the popularity model and Markov Chains-based models.

In the next three chapters, we will present our three cross-domain recommender system approaches designed to reduce the sparsity problem and improve the recommendation accuracy in the target domain.

Chapter 4

Cross-domain recommender system based on neural collaborative filtering

4.1 Introduction

With the fast-growing amount of information available in virtually every domain, the role of recommender systems (RS) has become crucial for many organizations and businesses. Specifically, personalized recommender systems are highly beneficial and more desirable than non-personalized ones because their outcomes are significantly more relevant to the interests of the end users (Cheng et al., 2016; Hadash et al., 2018). Collaborative Filtering approaches, which exploit existent user-item interactions, can be seen as the backbone of personalized recommender systems. Matrix Factorization (MF), a well-known CF approaches, applies inner products on user's and item's latent vectors to restore the missing ratings (Rendle et al., 2009).

Recently, employing Neural Networks in recommendation systems, a type of approach known as Neural Collaborative Filtering (NCF), has shown promising results. This line of work was inspired by the massive success of Deep Learning (DL) in Computer Vision (Yosinski et al., 2014), Speech recognition, and Neural Language Processing (Yang et al., 2017). The key idea of NCF is to replace the inner products of the MF approach by Multi-

Layer Perceptron (MLP) networks (Dziugaite and Roy, 2015; He et al., 2017b).

A main limitation of the above methods, MF and NCF, is that they both suffer from data sparsity and cold start problems when applied only on a single domain (Hu et al., 2018; Nguyen and Takasu, 2018). Usually, a small part of the interaction matrix is available while many other entries are missing, resulting in a sparse matrix. This can hinder the model's performance especially when the matrix is highly sparse (Hsieh et al., 2017; Nguyen and Takasu, 2018).

One effective alternative to tackle these problems is by transferring knowledge from other related domains, an approach generally referred to as cross-domain recommender systems. The cross-domain approach becomes a promising solution for improving recommendation accuracy in the target domain. People, in reality, interact with different items from different domains in daily life. This motivates us to build a model that can learn a better representation of a user based on multiple domains. Another motivation is that transferring knowledge from one or more related *source* domains to a *target* domain can alleviate the sparsity problem and improve the performance of the prediction model (Yang et al., 2017). Thus, we aim to build a cross-domain model that exploit information in these domains and transfer knowledge across domain to improve the performance of RS tasks.

Two types of feedback can be used as input for recommender systems: explicit feedback obtained directly from the user (e.g., ratings or like/dislike), and implicit feedback that is obtained by observing the interactions of the users with items (e.g., the number of clicks). In recent years, literature has been shifting from explicit feedback towards using implicit feedback as this is easier to obtain (He et al., 2017b; Hu et al., 2008).

Any two domains can overlap in terms of either users, items, both, or none (Cantador et al., 2015). This chapter focuses on transferring knowledge between domains that share users, in other words, we assume (almost) the same users in both domains but different items in each domain.

The major contributions of this chapter are as follows:

- We present a novel approach that extends the Neural Collaborative Filtering approach to a cross-domain recommendation setting.

- We develop a transfer learning approach to transfer knowledge from the source domain to the target domain to improve the prediction accuracy in target recommendation models.
- We perform extensive experiments on two real-world cross-domain datasets to investigate the effectiveness of applying NCF for cross-domain recommender systems. Furthermore, we compare our model with the state-of-the-art recommender system models.

The rest of the chapter is organized as follows: Section 4.2 describes state-of-the-art recommender systems. Section 4.3 describes the problem as well as the one-domain NCF network. In section 4.4, we introduce our proposed model. In section 4.5, we conduct a thorough experimental investigation of our model and compare it with several baseline models.

4.2 Related work

Most traditional RS models are build based on One-Domain CF (OD-CF) approaches, although Cross-Domain CF (CD-CF) approaches have become popular in recent years. In this section, we discuss the related work of both approaches.

4.2.1 Single-domain collaborative filtering

As mentioned above, Matrix Factorization (MF) is one of the most effective techniques of CF for predicting the missing user-item values. However, MF is unable to perform a recommendation for a user or an item with very few historical interaction records (Cheng et al., 2016; Dziugaite and Roy, 2015; Hu et al., 2018). To address this sparsity issue, MF was integrated with different models. For example, Koren et al. (2009) suggested incorporating user/item information within the MF model to learn about both user/item similarity. Furthermore, Li et al. (2010) suggested including user profile information in such MF models to address the sparsity problem. However, these attempts usually do not succeed because user profiles are often incomplete and highly sparse. Moreover, user profiles have some noise given that users give inaccurate information about their age, gender, and other metadata details. Thus,

user profiles can provide inaccurate representations of the users, which leads to irrelevant recommendations (Chiang et al., 2015).

Recently, DL have become a viable alternative and reported a better recommendation results. An early major approach has been proposed by Salakhutdinov et al. (2007), called Restricted Boltzmann Machines (RBMs). RBMs consists of two layers (a visible layer and a hidden layer) that model the explicit rating of a user on an item. This work was a ground base for many other works on missing rating approximation tasks.

More recent models combine a Neural Network model with another linear CF model, e.g., MF (He et al., 2017b), item similarity (He et al., 2018), Neighborhood model (Bai et al., 2017). This enables the model to be more powerful and expressive in learning user/item features by using linear and non-linear functions. Similarly, In (Cheng et al., 2016), the proposed model was a combination of two models, a linear model (Wide) and a non-linear model (Deep), which were jointly trained. This work concludes that the combined model shows relatively better results when compared to Wide-only and Deep-only models (Cheng et al., 2016).

Another relevant work was done by He et al. (2017b), called Neural Collaborative Filtering (NCF). Under NCF approach, MF was extended into a non-linear setting called Generalized Matrix Factorization (GMF). This was achieved by using unfixed weights and a non-linear activation function for the prediction layer. The same paper (He et al., 2017b) proposes an MLP model that feeds a concatenated user's and item's latent vector into multiple hidden layers. These two models can be trained separately or simultaneously. At the last stage, the combined model fuses the two individual models by concatenating their last hidden layer into one layer, called NeuMF.

Although DL empowers RS models, applying it only on one domain limits the performance of these models significantly. The sparsity problems still degrade the performance of OD-CF models. In consequence, the need for cross-domain models is crucial and has been recently widely recognized.

4.2.2 Cross-domain collaborative filtering

Cross-domain CF approach aims at exploiting knowledge from different source domains to improve the accuracy performance in the target domain. An early work based-MF approach to transfer knowledge between domains is called Collective Matrix Factorization (CMF) (Singh and Gordon, 2008). CMF factorizes two matrices simultaneously and allows knowledge transferring across domains by sharing user latent features. Pan et al. (2011) extends CMF to consider different latent factors of users. However, this approach has difficulty in learning complex user-item interactions due to the linear factorization of MF.

In another related work, Li et al. (2009a) construct a codebook transfer that gathers similar users and similar items into clusters. In this case, users/items in a cluster have a similar representation and can be utilized to transfer knowledge. He et al. (2019) extends the codebook transfer to automatically choose the codebook scale to transfer knowledge, that can be adaptive with the size and features of the source domain. Another recent model based on NCF is called cross collaborative network based cross domains (CoNet) (Hu et al., 2018). CoNet allows transferring knowledge between domains using two MLP models. It applies the cross-stitch network (Misra et al., 2016) to transfer knowledge from the source to the target domain and vice versa. It is worth mentioning that this model transfers knowledge while training. Also, Zhu et al. (2018) propose a deep framework for CD-CF based on MF and applies neural network to map latent vectors across domain.

In the following subsection, we emphasize some of the challenges of transfer knowledge across domains.

Knowledge transfer. Knowledge transfer, the base of any CD-CF model, is capable of alleviating the sparsity of one domain by leveraging knowledge from other domains (Hu et al., 2018; Li et al., 2009a). However, some of the challenges these algorithms face are considered to be related to what, when, and how to transfer. Indeed, each one of these questions is significantly important in increasing the effectiveness of transferring knowledge across domains. For instance, the information being transferred from the source domain should be consistent with the target domain (Lu et al., 2013). Otherwise, this transferring

may degrade the performance of the recommendation task. One key idea of transferring knowledge is by learning the latent factors of users and items in both domains and then using these learned user (item) latent vectors to transfer knowledge. This approach is beneficial especially under these conditions: a) if domains share users. Under the hypothesis that a user behaves similarly with different items from different domains, transferring learning between these domains is consistent and provides valuable information to the target domain. b) if the domains are related (e.g., Books and Movies). Furthermore, it has been proven that it is more beneficial to transfer knowledge at the early layers of the model where the information is general compared to the middle layers in deep stages where information is more specific (Yosinski et al., 2014).

4.3 Preliminaries

We first formalize the problem of cross-domain RS. Then, the one-domain neural CF network is described.

4.3.1 Problem description

This research focuses on decreasing the sparsity problem of one-domain recommendation by extending NCF model into a cross-domain recommendation setting. The high-level description of the problem is as follows:

Let T and S denote the target domain and the source domain, respectively. The goal is to improve the prediction accuracy in the target model by transferring knowledge from the source model. Both T and S share some users denoted by U of size m. The items in the two domains are different, and we denote the target items by I (of size n) and the source items by J (of size l). Each domain consists of implicit feedback of user-item interactions. For the target domain, we construct a binary matrix $D_T = \mathbb{R}^{m \times n}$ where each entry $d_{ui} = \{0, 1\}$. Likewise, we construct a binary matrix for the source domain $D_S = \mathbb{R}^{m \times l}$ where each entry $d_{ui} = \{0, 1\}$. Entry d_{ui} set to 1 if user u has interacted with item i (observed interaction)

and 0 otherwise (unobserved):

$$d_{ui} = \begin{cases} 1, \text{if user u has interacted with item i} \\ 0, \text{otherwise} \end{cases}$$

A user-item matrix is very sparse because a user usually interacts with a small subset of items. The task of a recommender system is to predict the score of unobserved entries. Then, based on the predicted scores, the recommender system ranks all items for each specific user. To predict the score for d_{ui} in one-domain CF:

$$d_{ui} = f(u, i|\theta) \tag{4.1}$$

where f denotes the interaction function and θ denotes model parameters.

For cross-domain CF, we intend to transfer knowledge from the source domain to the target domain. This is possible because the two domains share some users. Since every user $u \in U$ has interacted with some items $i \in I$ in the target domain and some items $j \in J$ in the source domain, we can exploit the source items to transfer knowledge across domain. Let $J^u = \{j_1, j_2, ..., j_x\}$ be the source items for a given user $u \in U$. Then, Equation 4.1 can be extended to include transferred information from the source domain by:

$$d_{ui} = f(u, i, J^u|\theta) \tag{4.2}$$

4.3.2 Single-domain neural CF network

The general structure of NCF network we apply is similar to the structure of the Deep model in (Cheng et al., 2016) and MLP in (He et al., 2017b). The general network takes a pair of user u and item i as the input, and feeds them through multiple layers to output the predicted score of d_{ui}. In the **input** side, the model applies the one-hot encoding method to map the user and item IDs into sparse vectors of all 0's except the ID's index position to be set to 1. Thus, the input feature representation of a user and an item will be $r^u \in \{0, 1\}^m$ and

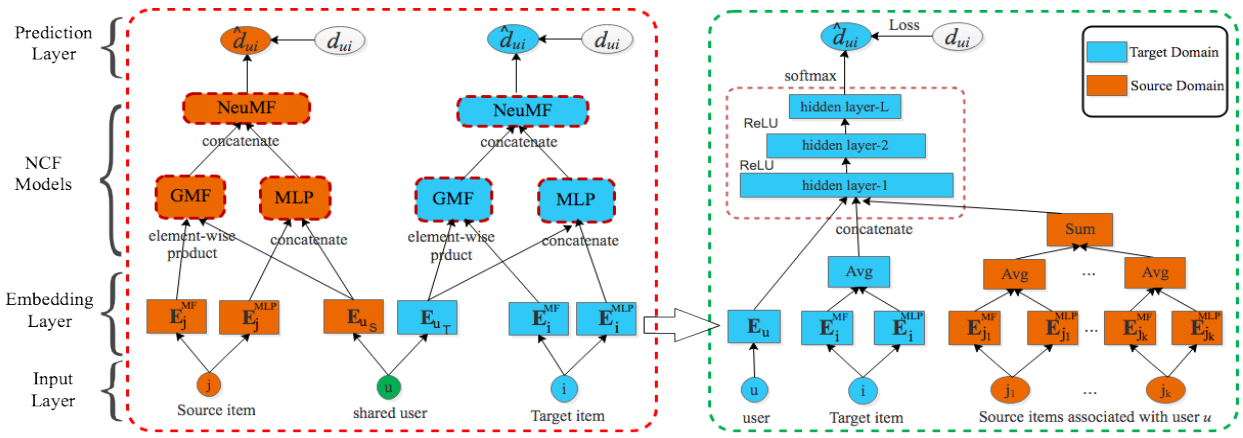


Figure 4.1: Cross-domain based on neural collaborative filtering

$r^i \in \{0,1\}^n$. In the **embedding layer**, the model takes each user's (item's) sparse vector and projects it into a continuous representation. Thus, the user's (item's) embeddings are in the form of $E_u = [P^T, r^u]$ and $E_i = [Q^T, r^i]$ where $P \in \mathbb{R}^{m \times d}$ and $Q \in \mathbb{R}^{n \times d}$. Then, both user's and item's embeddings are concatenated as $E_{ui} = [P^T, r^u, Q^T, r^i]$ and fed into the hidden layers (in this case, multi-layer perceprtons). In the **hidden layers**, the model feeds the concatenated embedding layer E_{ui} into multiple hidden layers of length L, where each hidden layer is utilized to uncover (different) features of user-item interactions. In the **output layer**, also called the prediction layer, the model takes the representation of the last hidden layer and predicts the score d_{ui} for each user-item pair. The output d_{ui} is constrained to the range of [0,1] by using a probabilistic function. The following function formulates the whole process:

$$f(E_{ui}|P, Q, \theta_f) = \phi_o(\phi_L(...\phi_2(\phi_1(E_{ui}))...)) \tag{4.3}$$

where ϕ_o and ϕ_L are the functions to compute the output layer and the hidden layers, respectively. Using the above equation the score d_{ui} can be seen as:

$$d_{ui} = f(E_{ui}|P, Q, \theta_f) \tag{4.4}$$

In the next section, we introduce our model for cross-domain recommender system using the Neural CF network.

4.4 The CD-NCF model

In this section, we first introduce the architecture of our model CD-NCF. Then, we discuss the model learning.

4.4.1 Model architecture

Our model focuses on leveraging information from the source domain to alleviate sparsity and increase accuracy in the target domain model. Because a user interacts with different items from different domains, we can utilize the learned item's embedding vectors to transfer knowledge across-domain as follows:

First, we train both domains (source domain and target domain) using NeuMF model as shown in the left side of Figure 4.1 (orange represents source and blue represents target). NeuMF combines GMF (perfrom an element-wise product) and MLP (a multi-layer feed-forward neural network) by concatenating their last layer to boost user-item interaction model by learning both models togather. It then outputs the prediction score d_{ui}. To provide more flexibility and expressiveness, the model learns distinct (independent) item embeddings as in He et al. (2017b) instead of sharing the embedding. For target item i, GMF and MLP learns separate embedding as E_i^{MF} and E_i^{MLP} (similarly for a source item j as E_j^{MF} and E_j^{MLP}). We denote E_u as user embedding. NeuMF is defined by the following equation:

$$l^{GMF} = (E_{u_T} \odot E_i^{MF}) \tag{4.5}$$

$$l^{MLP} = a_L(W_L^T(a_{L-1}(...a_2(W_2^T \begin{bmatrix} E_{u_T} \\ E_i^{MLP} \end{bmatrix} + b_2)...)) + b_L) \tag{4.6}$$

$$d_{ui} = \sigma(h^T \begin{bmatrix} l^{GMF} \\ l^{MLP} \end{bmatrix}) \tag{4.7}$$

where W, b, a are the weight matrix, the bias, and the widely used rectified activation

function (ReLU), respectively. h is the weight parameter learned from data.

After training finished, we save the learned latent vectors (embeddings) of users, target items, and source items and utilize them to transfer knowledge across domain.

Second, we train the target domain using MLP as shown on the right side of Figure 4.1. We initialize the model with the learned latent vectors from the previously trained model (NeuMF). Thus, the inputs include a user u, a target item i, and the source items associated with user u from the source domain. Specifically, we have the learned user's embedding, the learned target item's embeddings, and the learned source items' embeddings (items associated with a given user). Note, this step answers the questions about *what information to transfer* where the model transfers information from the user's source items into the target model. Also, the transfer happens at the early stage (at embedding layer) indicating *when to transfer* phase.

As mention above, each item has two learned embeddings, one from GMF and the other from MLP. For a target item, E_i^{MF} and E_i^{MLP}, we take the average of these embeddings as Avg^{E_i} to be the average weight embedding for target item i. Likewise, Avg^{E_j} is the average weight embedding for a source item j. In term of source items, we take the weighted sum of all these average embeddings of associated source items to transfer knowledge for a given user. Then, we concatenate all these learned embedding layers and feed them into the hidden layers. This step points out *how to transfer* phase happens.

We can formulate the input into the first (hidden) layer of MLP by

$$z_1 = \phi_1(E_{u_T}, Avg^{E_i}, \sum_{j=1}^{k} Avg^{E_j})$$

where E_{u_T} indicates user's learned embedding layer from the target domain. The second (hidden) layer of MLP is computed as

$$\phi_2(z_1) = a_2(W_2^T z_1 + b_2)$$

and so on. The last layer outputs the predicted score of d_{ui} that indicates how likely user u

is to interact with item i. In terms of item recommendation, the prediction scores are used to rank all unseen items for each user.

4.4.2 Model learning

Applying the squared loss function ($\hat{d}_{ui}$ - d_{ui})^2 to learn model parameters is not sufficient especially in collaborative filtering with implicit feedback. Indeed, the squared loss is applicable for rating prediction. To adequately address the nature of implicit feedback, where the prediction score is in the range of $[0, 1]$, we apply the following objective function called a binary cross-entropy loss:

$$L_f = - \sum_{(u,i) \in D^+ \cup D^-} d_{ui} \log \hat{d}_{ui} + (1 - d_{ui}) \log(1 - \hat{d}_{ui}) \tag{4.8}$$

where D^+ denotes the positive user-item interaction matrix and D^- denotes the sampled negative user-item interactions. This objective function can be seen as the negative logarithm of this likelihood function:

$$L(\theta) = \prod_{(u,i) \in D^+} \hat{d}_{ui} \prod_{(u,i) \in D^-} (1 - \hat{d}_{ui}) \tag{4.9}$$

The optimization of the objective function is done by using stochastic gradient descent (SGD) with the adaptive moment method (Adam).

Table 4.1: Datasets and domains' statistics

Dataset	#user	Target			Source		
		#item	#interaction	sparsity	#item	#interaction	sparsity
CiteULike	2177	15,699	58,469	99.83%	9,405	41,622	99.79%
Amazon	69,121	242,155	1,205,612	99.99%	42,751	730,915	99.97%

4.5 Experiments

We conduct extensive experiments to evaluate effectiveness of our model (CD-NCF) against the state-of-the-art recommendation models in terms of OD-CF and CD-CF with implicit feedback. In this section, we present the experimental setup and further analyze our results.

4.5.1 Experimental setup

Datasets. We utilize two real-world cross-domain datasets to evaluate our proposed model. The domains and statistics of both datasets are summarized in Table 4.1.

- **CiteULike** dataset[1]. This has been widely used by researchers since it is mostly based on research papers. Also, it is a very useful dataset in academia in various fields. We choose two domains. The target domain is *Paper-tag* that associates each paper with a number of tags to describe that paper. The source domain is *Paper-Citation* that specifies each paper and its citations. Our goal is to transfer information from the paper-citation domain to improve recommendation accuracy in the paper-tag domain (recommending new tags for each paper). We consider that paper cites another paper or paper is associated with a tag as positive interaction. Also, we filter out paper with less than ten interactions.

- **Amazon** dataset[2]. This has been utilized largely to evaluate collaborative filtering methods. We choose the two largest domains. The target domain is *Books* that consists of user-book ratings. The source domain is *Movies & TV* that consists of user-movie ratings. We consider 4-5 ratings as positive interactions. We aim to transfer information from the Movies domain to improve recommendation accuracy in the Books domain.

Evaluation protocol. In terms of item recommendation, we adopt the leave-one-out (LOO) method to evaluate the performance of our model. The LOO method is applied

[1]http://www.citeulike.org
[2]http://jmcauley.ucsd.edu/data/amazon/

widely in evaluating collaborative filtering models. We follow the protocol applied in (He et al., 2017b) and (Hu et al., 2018), that reserves the last interaction of each user for testing while training the model on the remaining. We follow the common approach that randomly samples a list of 99 negative items for each user and adds the test item to be the 100^{th} item. We then evaluate how good our model is to rank the test item at the top of the list. We further cut off the list at (N=10) to see if our model can rank the test item in the top-10 items. The evaluation matrices used to judge our model are Hit Ratio (HR) and Normalized Discounted Cumulative Gains (NDCG). HR intuitively measures whether the test item is present at the top-N items of the list. NDCG also considers the hit position of the top-N items by assigning higher scores if it is ranked higher in the top-N items.

Baseline. We compared our model with the baselines of one-domain and cross-domain CF models.

- **GMF:** Generalized Matrix Factorization is a one-domain collaborative filtering model that extends MF to a non-linear setting (He et al., 2017b).
- **MLP:** Multi-layer perceptron is a state-of-the-art one-domain CF model. It is a deep model and considered as the base model of NCF (He et al., 2017b).
- **NeuMF:** Neural Matrix Factorization is a combined model that combines MLP and GMF model by fusing the last layer of each model into a concatenated layer for prediction (He et al., 2017b).
- **MLP-Pretrain:** it is an MLP model initialized with *target* user's and item's learned embedding. We define this model to be similar to our model but without transferring knowledge from the source domain. This would judge the effectiveness of transferring knowledge of our model.
- **CoNet:** cross collaborative network based cross domains is a Cross Domain CF model. It combines two MLP models and applies cross connection unit to transfer knowledge from the source domain to target domain and vice versa (Hu et al., 2018). Since CoNet shows superior performance over CMF model, we do not report the results of CMF.

Table 4.2: Performance results comparison of different models on two public datasets

CiteULike dataset												
	GMF		MLP		NeuMF		CoNet		MLP-Pretrained		CD-NCF	
Factors	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
8	0.6396	0.4522	0.6376	0.4312	0.6518	0.4747	0.632	0.447	0.6674	0.4807	**0.6876**	**0.5035**
16	0.6505	0.4816	0.6376	0.4484	0.6651	0.4992	0.6577	0.4541	0.6812	0.5026	**0.7028**	**0.5237**
32	0.6642	0.498	0.644	0.4509	0.6688	0.5041	0.6789	0.4754	0.6843	0.5141	**0.7097**	**0.5384**
64	0.6711	0.5045	0.6532	0.4778	0.678	0.5104	0.6701	0.4899	0.6927	0.5238	**0.7129**	**0.5464**
Amazon dataset												
	GMF		MLP		NeuMF		CoNet		MLP-Pretrained		CD-NCF	
Factors	HR	NDCG	HR	NDCG	HR	NDCG	HR	NCDG	HR	NDCG	HR	NDCG
8	0.278	0.1617	0.2772	0.159	0.2936	0.1814	0.2815	0.164	0.3112	0.1921	**0.3293**	**0.2015**
16	0.295	0.1771	0.2834	0.1677	0.3011	0.1818	0.2936	0.1722	0.3188	0.1976	**0.3306**	**0.2162**
32	0.3021	0.1807	0.2909	0.1721	0.3121	0.1917	0.3045	0.1818	0.3268	0.208	**0.3404**	**0.2251**
64	0.3095	0.1816	0.2961	0.176	0.3186	0.1876	0.303	0.1855	0.337	0.2158	**0.3554**	**0.2337**

Implementation. For NeuMF and CoNet models, we use the code provided by their authors. Our methods are implemented using Keras with TensorFlow as a backend. The parameters of the model are initialized randomly with Gaussian $\mathcal{N}(0, 0.01)$. To optimize the model, we use Adam with learning rate as 0.001. The size of mini batch is set to 256. Also, for each positive interaction, we sample four negative interactions because it has been proven that the best negative sampling ratio is from 3 to 5 (He et al., 2017b). In term of embedding size (no. of factors), we evaluate our model with the following sizes [8, 16, 32, 64]. Note, applying large factors may overfit the model. Also, to prevent the MLP model from overfitting, we apply the dropout regularization method (dropout=0.5). For the deep neural architecture in our model, we apply the common strategy of halving the layer size of the next higher layer, known as a tower pattern. This is a common configuration of MLP network (the base model). In our case, we initialize the MLP with the learned user's and item's latent vectors that have been learned from training NeuMF. For determining hyper-parameters, one interaction for each user was randomly sampled as the validation set.

4.5.2 Performance analysis

In this section, we compare our model with other baseline models in terms of performance accuracy. The results of all models w.r.t. different numbers of factors are shown in Table

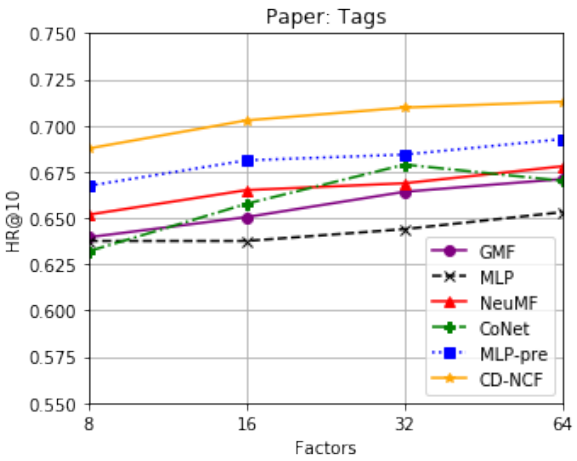
4.2. Our model shows better results when compared to one-domain CF models (GMF, MLP, NeuMF) and cross-domain CF (CoNet) on both datasets. To make it short and consistent, we discuss the results of the models with (number of factors = 32) although all results are reported on the table.

On CiteULike dataset, our model outperforms the base model (MLP) by 10% and 19% on HR and NDCG, respectively. Also, our model shows an improvement by at least 6% in term of HR and NDCG compared to GMF and NeuMF. Since these are one-domain models, this illustrates the benefits of transferring knowledge across domains. In terms of CoNet, our model surpasses it by 5% on HR and 13% on NDCG.

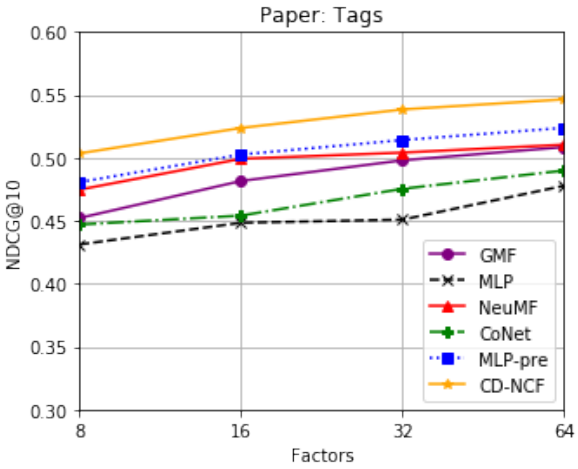
On Amazon dataset, where the Book dataset is extremely sparse, our model achieves better results than MLP and increases the performance by 17% and 30% on HR and NDCG. Also, our model surpasses GMF by 13% and 25%. Furthermore, the proposed model outperforms NeuMF and CoNet models by at least 10% and 15% regarding HR and NDCG. Figure 4.2 indicates that our model consistently shows better results (w.r.t. different factors) over all these recent existing models.

4.5.3 Transfer versus no-transfer

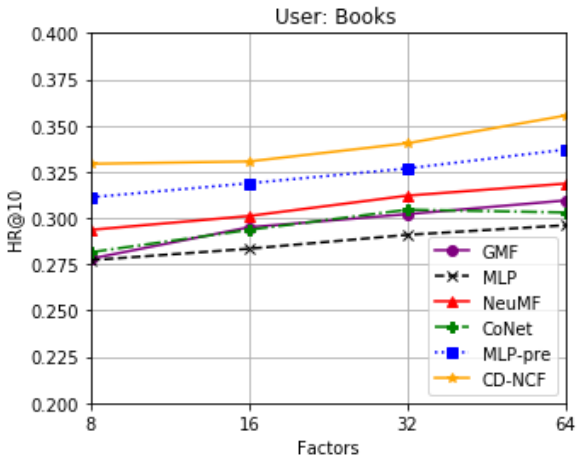
We further investigate the effectiveness of transfer knowledge by comparing our model with MLP-pretrain. MLP-pretrain is exactly like our model but without the transferring layers that transfer information from the source domain to the target domain. In CiteULike dataset, our model achieves an improvement by 5% on both HR and NDCG over MLP-pretrian. In Amazon dataset, also, the performance increase by 4% and 8% in HR and NDCG. Indeed, our model provides consistent improvement over MLP-pretrain with all different factors. This strongly indicates that transferring knowledge from a source domain is more beneficial than relying solely on one domain in recommendation tasks.



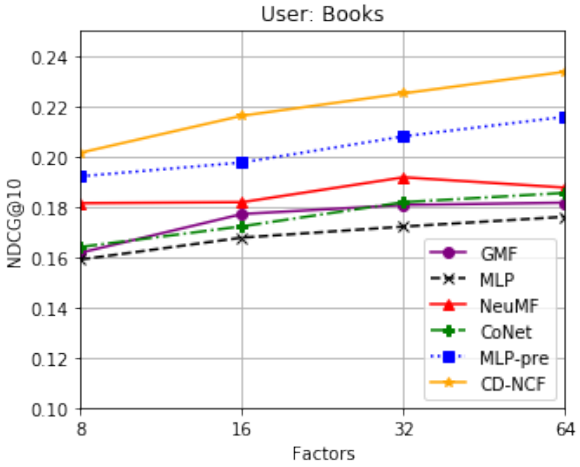
(a) CiteULike (HR@10)



(b) CiteULike (NDCG@10)



(c) Amazon (HR@10)



(d) Amazon (NDCG@10)

Figure 4.2: Performance Results of HR@10 and NDCG@10 on different factors on both datasets

4.5.4 Loss training analysis

Transferring knowledge using the learned latent vectors (embeddings) gives the model a good start at a low loss point. The training loss starts at a low point in the deep model because these users (items) embeddings have been trained using NeuMF and each user (item) has a better vector representation that describe a user (an item). On the other hand, transferring while training approach (e.g. CoNet) starts transferring information early where users (items) vectors do not have an accurate representation yet (before convergence). This

may transfer inaccurate information at the beginning of the training that may degrade the model performance. Our model provides consistent improvement over the CoNet model.

4.6 Summary

In this chapter, we proposed the (CD-NCF) model that extends the neural collaborative filtering approach to a cross-domain recommendation setting, and achieves effective knowledge transfer between the source and target domains. By transferring knowledge across domain, our model alleviates the impact of the data sparsity problem and improves the performance of the recommendation model in the target domain. We evaluated the proposed model on two real-world datasets and showed an improvement over the baseline models in both one-domain and cross-domain CF. Also, experiments showed the effectiveness of knowledge transfer over no knowledge transfer between the source and the target domains. Extensions of the CD-NCF model to include item's content and additional information about user-item interactions (e.g., user's reviews) will be investigated in future work.

Chapter 5

Cross-domain self-attentive sequential recommendations

5.1 Introduction

Sequential recommendation (SR) systems have received significant attention in recent years, due to their ability to model temporal information and capture a user's different preferences over time, ability that generally translates into an increased recommendation accuracy (Fang et al., 2019; Kang and McAuley, 2018). Specifically, SR systems model the user's historical behavior (e.g., items that the user has interacted with) in a sequential order, and predict the next behavior the user will exhibit (e.g., the next item the user will interact with). Like most popular single-domain recommender systems, SR models depend heavily on the user's historical behavior data (which can be either *explicit*, e.g., item ratings, or *implicit*, e.g., inferred from information about clicking, sharing, or buying an item) to generate personalized recommendations. For users with a limited behavior history (i.e., a small number of interactions), single-domain recommender systems perform poorly, suffering from the data sparsity and cold start problems (Koren et al., 2009).

Cross-Domain Recommendation (CDR) systems have been proposed to tackle the sparsity issue by transferring knowledge from a potentially dense source domain to a sparse target

domain. The CDR systems have become more prevalent in recent years due to the spread of e-commerce systems and the diversity of their products. For example, in Amazon, we can recommend new movies to a user by not only learning user preferences in the movie domain, but also transferring knowledge from the book domain (in the form of books that the users have read before) to increase the recommendation accuracy. Due to these reasons, shifting toward CDR models is highly desirable and necessary.

Thus, in an effort to overcome the sparsity issue, different CDR approaches have been introduced. Some early works have focused on utilizing traditional techniques. For example, Gao et al. (2013); Li et al. (2009b) group similar users and similar items into clusters, and transfer knowledge across domains at cluster level, by utilizing the clusters' latent representations. Others apply different variants of Matrix Factorization (MF) (Li et al., 2009a; Loni et al., 2014; Mirbakhsh and Ling, 2015; Singh and Gordon, 2008) to perform knowledge transfer. Due to their linear functionality, however, clustering and MF methods cannot capture complex features of user-item interactions. Equipped with Deep learning (DL) strategies, other works have utilized and extended models based on Multi-Layer Perceptron (MLP) (Hu et al., 2018) and Auto-encoders to transfer knowledge across domains. However, these non-sequential models can only learn static user preferences (Fang et al., 2019). Also, they assume that all user-item behaviors in the sequence are equally important (Fang et al., 2019).

There are two important challenges that need to be addressed in order to generate accurate cross-domain sequential recommendations. The first challenge is related to how to capture user's different preferences in the user's historical behaviours sequence. The second challenge is about how to weigh the importance of these behaviors in the sequence to aggregate an accurate representation vector (that can be used to transfer knowledge). The attention mechanism is a technique used to focus only on relevant parts of a sequence when calculating the output. Inspired by the Transformer (Vaswani et al., 2017), recently, Kang and McAuley (2018) proposed a state-of-the-art single-domain sequential recommendation model using the self-attention mechanisms (their model is called, SASRec). SASRec exhibits up to 10x speed improvements, when compared to previous CNN/RNN-based approaches. Nonetheless, single domain recommendation methods can suffer from data sparsity as they

disregard user's various behaviors across domains.

To overcome the outlined problems of single-domain recommendation models, we propose a Cross-Domain Self-Attentive Sequential Recommendation Model (CD-SASRec) that extends the state-of-the-art SASRec model to the cross-domain setting and endows the model with the ability to capture user's various preferences across domains. In CDR tasks, domains can be categorized based on user overlap, item overlap, both, or none. In this chapter, we rely on user overlaps, i.e., we require users to be shared between the source and target domain, while the items are assumed to be different, thus reflecting a real-world setting.

The main contributions of this chapter can be summarised as follows:

1. We propose a Cross-Domain Self-Attentive Sequential Recommendation model to alleviate the data sparsity of a single domain. To the best of our knowledge, this is the first cross-domain model for sequential recommendations.
2. We utilize the self-attention mechanism as our transfer knowledge approach, to weigh the importance of the items in the source domain and to transfer this knowledge to the target domain.
3. We perform extensive experiments on three real-world cross-domain datasets to investigate the effectiveness of applying CDR on sequential recommendation using self-attention mechanism. Furthermore, we compare our model with state-of-the-art sequential recommendation models.

The remaining of the chapter is organized as follows: we review related work in Section 5.2. The problem addressed is formally described in Section 5.3, where the approaches are also presented. Section 5.4 describes the experimental setup. We discuss the results in Section 5.5, and conclude the chapter in Section 5.6.

5.2 Related work

In this section, we first highlight the most popular single-domain sequential recommendation models. Then, we discuss existing CDR models.

5.2.1 Sequential recommendation

Most existing sequential recommendation models have been proposed for solving single-domain recommendation tasks. Early SR models adopt Markov Chains (MC), where the prediction of future user's behaviors depends only on the last or the last few interactions. These types of models either adopt first-order MC methods (He et al., 2017a) or, sometimes, high-order MC methods that include more previous behaviors (He and McAuley, 2016a; He et al., 2016a). Considering only few behaviors, MC-based models cannot captures a user's dynamic preferences. Another line of work, adopts models based on Convolution Neural Networks (CNN) and Recurrent Neural Networks (RNN). For example, Caser embeds the previous n items as an image, and then applies a convolutional filter to extract local features (Tang and Wang, 2018). GRU4Rec (Hidasi et al., 2015) is the first RNN model to address session-based recommendation, and makes use of a Gated Recurrent Units (GRU) model to distill the information form the previous step together with the current action to output (i.e., to predict) the next item. Hidasi and Karatzoglou (2018) improve the GRU4Rec model further in terms of Top-N recommendation performance. The downside of the RNN models is that they are unable to fully captures preferences of users with a very long behavior sequence. Recently, the attention mechanism has contributed greatly to improvements in sequential recommendation accuracy, and a variety of single-domain models have been proposed in the field (Kang and McAuley, 2018; Li et al., 2017; Lv et al., 2019; Zhang et al., 2018). Noticeably, SASRec (Kang and McAuley, 2018), a model that uses self-attention, has become the-state-of-the-art approach for sequential recommendation. Specifically, SASRec applies a self-attention layer over a sequence of items to predict the next item in the sequence. Nonetheless, the models reviewed here, including SASRec, are limited to only single domain recommendation tasks and do not account for users' preferences in other domains. As a consequence, the performance of the single domain recommendation models can be significantly affected by data sparsity.

5.2.2 Cross-domain recommendation

Previous cross-domain recommendation methods vary in terms of how they extract user's preference in the source domain, and how they transfer this knowledge to generate accurate recommendation in the target domain. Early CDR approaches focused on neighborhood-based models. MF-based and clustering-based approaches were dominant in the field for a long time. For instance, an approach called Collective Matrix Factorization (CMF) (Singh and Gordon, 2008) factorizes two matrices simultaneously and allows knowledge transfer across domains by sharing user latent features. Later on, Li et al. (2009a) designed a codebook-based transfer approach by grouping similar users and similar items into clusters. Subsequently, as users/items in a cluster have a similar representation, they can be utilized to transfer knowledge. Pan et al. (2011) extended the CMF approach to consider different latent factors of users. However, these models cannot capture user-item complex features because of their linear functions.

In term of deep learning, almost all previous CDR approaches model user-item interaction in a non-sequential manner. CoNet (Hu et al., 2018) utilizes two MLP models, and applies the cross-stitch network (Misra et al., 2016) to transfer knowledge between domains. However, these models cannot capture the dynamic changes in user's preferences over time because they consider a non-sequential approach. Non-sequential models assume that all user-item interactions are equally important.

In this work, we aim to fill in the gap between sequential recommendations and cross-domain recommendations. Therefore, we propose a model that can transfer knowledge across domains by considering user-item behaviors in a sequential manner (adding time dimension information), and by weighing and identifying important behaviors using the self-attention mechanism.

5.3 Problem description and approaches

In this section, we provide the problem description and introduce approaches.

5.3.1 Problem description

Our model concentrates on the problem of cross-domain sequential recommendation, assuming the source-target user overlap scenario, which reflects the real world setting where a user interacts with multiple domains. Therefore, we have a *source* domain and a *target* domain, denoted by S and T, respectively. The total number of source items is denoted by l, while the total number of target items is denoted by k. U represents the set of shared users. Each $u \in U$ has interacted with a subset of m source items, and a subset of n target items. We denote the item sequence of each user u in source and target by S_u and T_u, respectively. Formally, we have: $S_u = [s_{i_1}, s_{i_2}, \cdots, s_{i_m}]$, $T_u = [t_{i_1}, t_{i_2}, \cdots, t_{i_n}]$ The users' source and target sequences of items are listed in temporal order (from old to new). Our goal is to predict the next item $t_{i_{n+1}}$ for user u in the target domain by transferring user's preferences from the source domain.

5.3.2 Self-attentive sequential recommendation

In this section, we summarize the Self-Attentive Sequential Recommendation model (SASRec), proposed by Kang and McAuley (2018). SASRec applies the self-attention mechanism over a sequence of items, so that the next item is predicted by weighing the importance of the previous m items. Specifically, SASRec consists of an embedding layer, self-attention blocks, and a prediction layer. An illustration of the SASRec model can be seen on the left side of Figure 5.1 (where the model is shown for the source domain). Accordingly, we use the source domain to describe the SASRec model for single domain sequential recommendation.

Embedding layer: At the bottom of the model, there is an embedding layer that takes as input the item sequence and projects each items in the sequence into a low level dimensional vector. Specifically, the embedding layer, first, constructs a learnable item embedding matrix (for source items), $L \in \mathbb{R}^{l \times d}$, where d is the latent embedding dimensionality, and l is the total number of source items. The input of each user u, consisting of a sequence of m items, $S_u \in \mathbb{R}^m$, can be represented using an embedding matrix $E_u^s \in \mathbb{R}^{m \times d}$. In order for the self-attention layer to keep track of the temporal order of the item in the sequence,

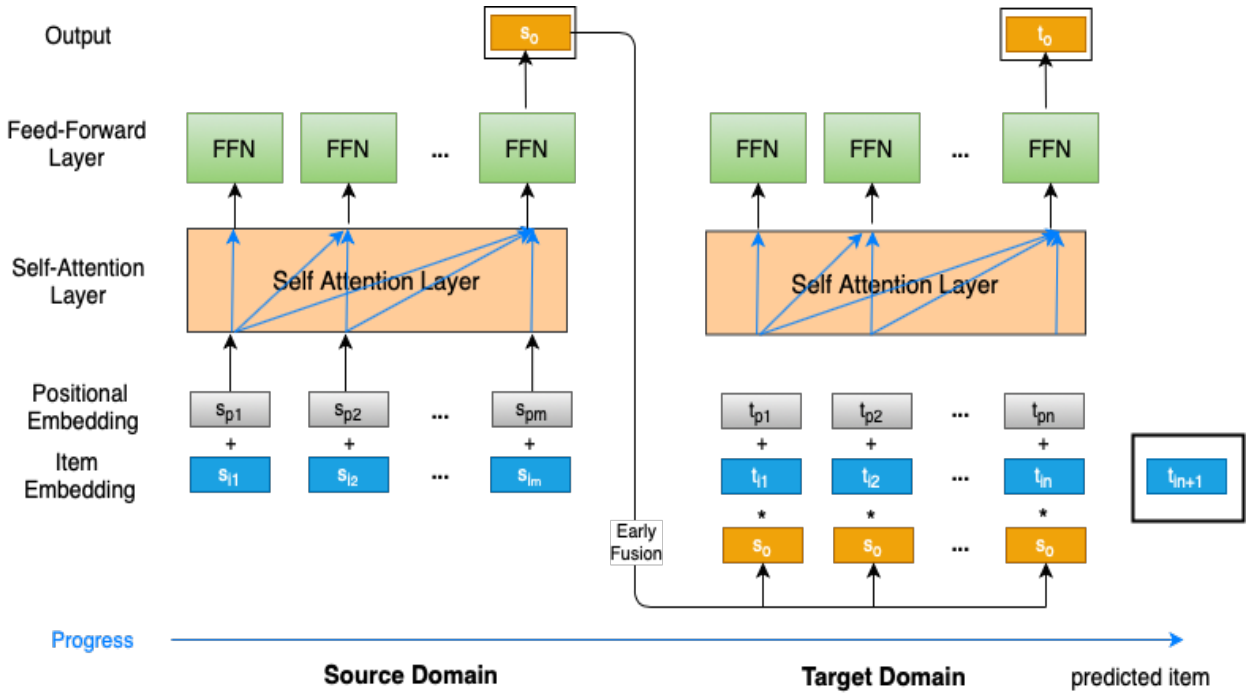


Figure 5.1: Cross-domain self-attentive sequential recommendation

a positional embedding layer $P \in \mathbb{R}^{m \times d}$ is added to the input embedding layer. Thus, the output of the embedding layer $\hat{E}_u^s$ is as follows:

$$\hat{E}_u^s = \begin{bmatrix} [s_{i_1} + s_{p_1}], \\ [s_{i_2} + s_{p_2}], \\ \cdots, \\ [s_{i_m} + s_{p_m}] \end{bmatrix}$$

Self-attention layer: The embedding matrix $\hat{E}_u^s$ is then fed to the self-attention layer, specifically, the scaled dot-product attention, defined in Vaswani et al. (2017) as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

where Q, K, V represents query, key, and value representations of each item. The inner product between the query and key representations is scaled by $\sqrt{d}$ to avoid large values, potentially resulting from the high dimensionality. More specifically, in SASRec, the query,

key and value representations of an item are obtained by projecting the embedding matrix $\hat{E}_u^s$ onto three matrices $W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$ (which need to be learned). These projections are subsequently fed into the attention layer: $s_{att} = Attention(\hat{E}_u^s W^Q, \hat{E}_u^s W^K, \hat{E}_u^s W^V)$. In the sequential recommendation model, the model aggregates links from the previous m embedded items with adaptive weights to predict the $m+1$ item.

Feed-forward layer: The output of the self attention layer s_{att} is then passed through two fully connected layers to endow the model with non-linearity and learn more complex features of the user-item interactions: $s_o = \phi(s_{att} W^{(1)} + b^{(1)}) W^{(2)} + b^{(2)}$, where $W^{(1)}, W^{(2)}$ are $d \times d$ matrices, $b^{(2)}, b^{(2)}$ are $d-$dimensional bias vectors, and ϕ is a ReLU activation function. Note that, SASRec can apply more self-attention and feed-forward layers to capture more deep item transitions. Inspired by Vaswani et al. (2017), the model applies residual connection to assist propagating low-level features into higher layers.

Prediction layer: The model predicts the next most relevant item, using the following equation: $r_{i,f} = \sigma(s_o \cdot s_{i_f})$, where σ is the sigmoid function, and s_o represents the output of the feed-forward layer at a timestamp f. In other words, s_o represents an aggregated vector based on the previous m items.

5.3.3 Proposed model

In this section, we describe the proposed Cross-Domain Self-Attentive Sequential Recommendation model (CD-SASRec). Our model aims to alleviate the sparsity issue of the single-domain recommendation model, SASRec. In the cross-domain setting, there is at least one *source* domain in which users have a large number of interactions with items, in addition to a *target* domain in which users may have very few interactions, not enough to generate accurate, personalized recommendations in the single-domain setting. Thus, our goal is to extend the SASRec model into to the cross-domain setting which can leverage more user preferences and complex item features from the source domain, and then transfer the aggregated knowledge to the target domain.

As mentioned above, each user u has interacted with a set of source items, S_u, and with

a small set of target items T_u. Here, we construct a source item embedding matrix $L \in \mathbb{R}^{l \times d}$ and a target item embedding matrix $K \in \mathbb{R}^{k \times d}$. For a user u, the source input sequence $S_u \in \mathbb{R}^m$ and the target input sequence $T_u \in \mathbb{R}^n$ can be embedded using the L and K embedding matrices.

CD-SASRec model addresses the task of performing cross-domain sequential recommendations using two main steps. The first step is learning and aggregating users' preference in the source domain. The second step is transferring these aggregated preferences into the target domain to enable the model to overcome the sparsity issue. The details of our model are as follow: First, we run the SASRec model on the source domain to aggregate information about a user u, using the self-attention layers, which produce a weighted sum of the embeddings of the previous source items in the input sequence S_u. The output of source model s_o can be seen as an aggregated vector that reflects user preferences in the source domain. After extracting user's preferences from the source domain, the main challenge is how to transfer the source knowledge the target domain in a way that effectively increase the target domain recommendation accuracy. Our approach is to fuse the aggregated source vector s_o (corresponding to user u) into the embedding vectors of the items in the target sequence of the user u, as illustrated on the right side of Figure 5.1. This method is usually referred to as an early representation fusion (Adler et al., 2020). It is a concept of incorporating information learned from the source model into the embedding layer at the target model. Intuitively, fusing the source information "early" as opposed to "late" (i.e., an upper layer in the target model), it enables the target to refine the source information and retain only the relevant pieces. Furthermore, transferring information at the early stages (e.g., embedding layer) is more advantageous compared to transferring at higher stages in the deep models where information is more specific (Yosinski et al., 2014). Thus, the embedding matrix of user u in the target domain is defined as follows:

$$\hat{E}_u^t = \begin{bmatrix} [s_o + t_{i_1} + s_{p_1}], \\ [s_o + t_{i_2} + s_{p_2}], \\ \cdots, \\ [s_o + t_{i_n} + s_{p_n}] \end{bmatrix}$$

Then, the combined embedding $\hat{E}_u^t$ is passed into multiple self-attention and feed-forward layers to predict the next item for user u in the target domain (e.g., through another SASRec model). One benefit of the transferred aggregated vector s_o of the user u is that it is helping the target model to start in a good position in terms of learning about the user u, despite the very small set of items that the user has interacted with in the target domain. Lastly, the prediction layer is using matrix factorization to calculate the relevance score of the next target item $t_{i_{n+1}}$ given the sequence of first n item and the transferred knowledge vectors: $r_{i,f} = \sigma(T_o \cdot [s_o, t_{i_f}])$. To alleviate over-fitting problem, especially due to having many layers (and thus many parameters), the model uses the dropout regularization technique, a widely applied regularization method. During training, for each a positive item $t_i \in T_u$, a negative item $j \notin T_u$ is uniformly sampled at each timestamp f. Thus, the loss function that the model optimizes is the binary cross-entropy loss defined as:

$$\sum_{t_i \in T} \sum_{f=[1,2,\ldots,n]} \sum_{j \notin t_i} -[\log(r_{i,f}) + \log(1 - r_{j,f})]$$

5.4 Experimental setup

The experiments are design to investigate the effectiveness of our knowledge transfer approach for sequential recommendation and to compare it with several sequential recommendation models.

5.4.1 Dataset

We utilize the existing Amazon dataset (He and McAuley, 2016b) by extracting multiple subsets with shared users in different categories (a.k.a., domains). Specifically, we focus on the three largest and most related categories, *Books*, *Movies*, and *Music*. We construct and experiment with different pairs of source-target domains. We select users with at least 10 interactions in each domain. Our goal is to transfer knowledge from the source domain (e.g., *Books*) to alleviate the sparsity problem in the target domain (e.g., *Movie*) and, thus, improve the overall recommendation accuracy in the target domain. Data statistics are summarized in Table 5.1.

Following (He and McAuley, 2016a; He et al., 2017a; Kang and McAuley, 2018; Rendle et al., 2012), we use the existing ratings as implicit feedback and convert them to binary values (i.e., 1 for interaction and 0 for no interaction). The sequence order is determined by the actual timestamp. We use the standard *leave-one-out* approach to split our data for training and testing purposes. The last item is reserved for testing, the second-to-last for validation, and the remaining items are used for training.

To effectively evaluate our model against the data sparsity, we adopt three scenarios for user interactions in the target domains. The first one is to include all users interactions in the target domain. The second one is to include only the first 10 interactions. The last scenarios is to include only the first 5 interactions for each user in the target to make the target domain even sparser, and the task for our model more difficult. For example, in the last scenario that assumes only 5 items/interactions, we use only three items for training, while one is used for validation, and the last one for testing. In all three scenarios, however, we include all user interactions from the source domain to understand the effectiveness of our knowledge transfer approach.

5.4.2 Evaluation metrics

To evaluate our model performance, we utilize two popular ranking metrics, Hit Ratio at 10 (HR@10) and Normalized Discounted Cumulative Gain at 10 (NDCG@10), for top-N

Table 5.1: Statistics on the **source→target** pairs of Amazon domains considered in our evaluation, including the number of users by the domains in a pair, and also the specific numbers of items and interactions in each domain in the pair.

Source→Target pair	**Book → Movie**		**Book → Music**		**Movie → Music**	
# users	9,482		4,427		5,465	
Domain	Book	Movie	Book	Music	Movie	Music
# items	184,965	41,503	122,477	50,012	38,091	53,355
# interactions	632,678	421,204	308,525	201,315	310,672	268,628

recommendations. HR@10 calculates the fraction of cases/hits in which the test item is present in the top-10 items, while NDCG@10 considers also the hit position by assigning higher scores if the item is ranked higher in the list of the top-10 items. With large datasets, it is computationally expensive to rank all items for each single user. To avoid massive computations, we adopted the strategy used in (He et al., 2017b; Koren, 2008). For each user, we randomly sample a list of 99 negative items and we add the test item to the list. After that, our model ranks these 100 items based on the predicted score for each item. Ideally, the test item would be ranked higher in the list.

5.4.3 Baselines

We evaluated the **CD-SASRec** model against several baselines:

- **Pop:** This is a simple non-personalized model that performs ranking based on item popularity (i.e., number of interactions associated with each item).
- **Bayesian Personalized Ranking (BPR-MF):** This is a classic personalized ranking method for implicit feedback, coupled with MF (Rendle et al., 2012).
- **Factorized Personalized Markov Chains (FPMC):** This baseline integrates MF with factorized first order Markov Chains (Rendle et al., 2010).
- **Self-Attentive Sequential Recommendation (SASRec):** This is a state-of-the-art sequential recommendation model (Kang and McAuley, 2018) (as described in

Section 5.3.2).

Note that some existing sequential models (e.g., GRU4Rec (Hidasi et al., 2015) and Caser(Tang and Wang, 2018)) were not included as baselines, as Kang and McAuley (2018) have already shown that the SASRec model outperforms these types of single models significantly. Also, we did not include existing cross-domain models as they are mainly non-sequential, while our model is designed for sequential tasks.

5.4.4 Model setting

We implement our proposed model, CD-SASRec, and the baselines, BMR-MF and FPMC, using TensorFlow, and we utilized the code provided by the authors for SASRec. All baselines are optimized with Adam (Kingma and Ba, 2014), and the hidden layer size is set to 50. We tune hyper-parameters using the validation set.

For our model, we follow the architecture of SASRec by applying multiple self-attention and hidden layers for both domains. The batch size and learning rate are set to 128 and 0.001, respectively. The dropout strategy is applied to prevent over-fitting, and fine-tuned using the following rates $0.5, 0.6, 0.7, 0.8, 0.9$. We found that the best rates are 0.5 for the source domain and 0.8 for the target domain. The sequence length is set to 50 for the source domain, and it depends on the scenario employed for the target domain (i.e., 50, 10, 5 items, respectively). We pad the sequence at the positions corresponding to the oldest items, if it has less than the max number of the items.

5.5 Results and discussion

In this section, we present the results of our cross-domain model by comparison with the results of the baselines, and also discuss the overall performance of the model. We investigate three different scenarios for the source→target pairs considered: specifically, all, only 10, or only 5 items, respectively, are assumed available for the users in the target domain. The goal is to gain insights into the effectiveness of our knowledge transfer approach in relation

to data sparsity. The results of our model and baselines are presented in Tables 5.2, 5.3, 5.4 for pairs Book→Movie, Book→Music, and Movie→Music, respectively.

The results show that our model consistently outperforms all the baselines within the most difficult scenario, when only 5 target items are available for a user. Remarkably, for the Book→Movie pair, our model surpasses SASRec by 7% and 17% in HR@10 and NDCG@10, respectively. The results illustrate the effectiveness of transferring knowledge from the source domain when users have very few interactions in the target domain.

In the second scenario, when only 10 items are available for a user in the target domain, our model performs significantly better than the Pop, BPR-MF, FPMC baselines for all pairs. It also performs better than the SASRec model for the Book→Music and Movie→Music pairs, while the results are very close to those of the SASRec model for the pair Book→Movie, which exhibits the largest number of interactions per item in the target domain.

In the last scenario, where all the target interactions are used, our model outperforms the Pop, BPR-MF, FPMC baselines for all pairs. Surprisingly, it also slightly outperforms the SASRec model for Book→Movie pair, while its results slightly worse than those of the SASRec model for the and Movie→Music and Book→Movie pairs. Together the results of our experiments suggest that the CD-SASRec achieves significantly better performance when the user-item interactions are sparse in the target domain. As the number of interactions increases, the benefits observed from transferring knowledge from the source diminish, and the knowledge transferred can even become a source of noise.

Table 5.2: Performance results for the Book→Movie pair

Source→Target Pair	Book → Movie					
Scenario	*all*		$items = 10$		$items = 5$	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Pop	0.4381	0.2583	0.2259	0.1129	0.1745	0.0897
BPR-MF	0.4103	0.2406	0.3111	0.1769	0.2265	0.1271
FPMC	0.4734	0.2862	0.3879	0.2329	0.2433	0.1364
SASRec	0.6153	0.4111	**0.4777**	**0.2898**	0.3923	0.2061
CD-SASRec	**0.6297**	**0.4154**	0.4742	0.2823	**0.4208**	**0.2403**

Table 5.3: Performance results for the Book→Music pair

Source→Target Pair	Book → Music					
Scenario	*all*		$items = 10$		$items = 5$	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Pop	0.2974	0.1443	0.1778	0.0921	0.1195	0.0603
BPR-MF	0.2821	0.1644	0.1965	0.1088	0.152	0.0759
FPMC	0.3566	0.2171	0.2062	0.1131	0.1633	0.0991
SASRec	**0.4276**	**0.271**	0.2573	0.1392	0.2306	0.1135
CD-SASRec	0.423	0.2685	**0.2697**	**0.1506**	**0.2365**	**0.1202**

5.6 Summary

In this chapter, we proposed a cross-domain self-attentive sequential recommendation approach (CD-SASRec) that models user temporal information in a pair of source and target domains to extract user preferences from the source domain and transfer it into the target domain. We applied multiple self-attention and feed-forward layers over the embeddings of the items in the source sequences of a user to learn user source-representational vector. The vector was then used to transfer knowledge about the user into the target domain by combining it into the embedding layer of the target component of the model. We evaluated our model on three large datasets using three different numbers of interactions. Experimental results showed that our cross-domain model outperforms all the baselines considered when

Table 5.4: Performance results for the Movie→Music pair

Source→Target Pair	Movie → Music					
Scenario	*all*		$items = 10$		$items = 5$	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Pop	0.3211	0.1835	0.1605	0.0942	0.1347	0.0732
BPR-MF	0.3392	0.2019	0.2177	0.1241	0.1711	0.0976
FPMC	0.4274	0.2597	0.2207	0.1282	0.1881	0.1015
SASRec	**0.5074**	**0.3268**	0.2887	0.1588	0.2699	0.1372
CD-SASRec	0.5048	0.3181	**0.3107**	**0.1753**	**0.2765**	**0.1485**

including very few user interactions, and gives results similar to those of the SASRec model when a larger number of interactions are available for users in the target domain.

Chapter 6

Cross-domain attentive sequential recommendations based on general and current user preferences

6.1 Introduction

Recommender Systems (RS) have become an essential tool to extract useful information about users to alleviate the problem of information overload, thus saving user's time and effort from facing a large (numerous) number of choices (Hu et al., 2017; Lian et al., 2017; Wang et al., 2018). Traditional Recommender System models deal with the user-item interactions as a set of pairs, to extract the overall user's general (static) preferences by utilizing the learned low dimensional vectors of users and items (Hu et al., 2008; Kumar et al., 2017). Unlike traditional models, sequential Recommender System models consider user's historical behaviors as an ordered sequence of items to learn the user's current (dynamic) preferences over time to closely predict the next item the user will engage with (Liu et al., 2018; Ying et al., 2018). Recently, the sequential Recommender System models have attracted significant attention because they reflect the real-world setting where a user interacts with items over time.

Lately, the development of sequential Recommender System models is centered around using deep learning methods, such as Recurrent Neural Network (RNN) (Donkers et al., 2017; Hidasi and Karatzoglou, 2018) and Convolution Neural Networks (CNN) (Tang and Wang, 2018; Yuan et al., 2019). More recently, the trend of this development has shifted toward using the attention mechanism (Kang and McAuley, 2018; Ying et al., 2018) to overcome the vanishing gradient problem encountering for very long sequences. The attention mechanism assigns importance weights to the items in the sequence and makes the next decision based on the aggregation of all weighted (important) items. SASRec (Kang and McAuley, 2018) is a state-of-the-art sequential recommender system that captures the user's long-term and short-term preferences by using series of self-attention layers. It is worth noting that more user-item interactions help the sequential models to be more informative and accurately predict the next item.

By benefiting from learning user's static and dynamic preferences, recent works (He et al., 2020; Lin et al., 2020) have proven to be more effective in term of recommendation accuracy, when combining multiple user preferences models for a single-domain task. For example, CAR (He et al., 2020) and FISSA (Lin et al., 2020) combine two (different) user preference models, where one model is set to capture the user's overall general taste, while the other one is set to capture the user's dynamic preferences. Then, they fuse aggregated preferences to make predictions. Despite all these improvements, single-domain models face a real challenge when users have very few historical records and the system cannot extract meaningful information about these users, a problem referred to as data sparsity. Clearly, the data sparsity makes it difficult for these single models to predict the next user's behaviors, or even to aggregate a good user representation, and as a consequence produce inaccurate recommendations (Hu et al., 2018, 2019; Man et al., 2017; Sahu and Dwivedi, 2019).

Cross-Domain Recommender (CDR) systems have been proposed to tackle the sparsity of user-item interactions by transferring information from related (dense) source domains into the target domain to increase prediction accuracy (Lian et al., 2017; Zhong et al., 2020). For example, Amazon has the Book and Movie domains, which share some users. The information learned about a user from the Book domain can be utilize to make more

reasonable (satisfactory) prediction in the Movie domain. Most existing CDR models have only focused on learning users' general preferences utilizing mainly the Matrix Factorization (MF) approach (Huang et al., 2019; Man et al., 2017; Sahu and Dwivedi, 2018, 2020), which cannot capture the non-linearity of user-item interactions. Furthermore, the CDR relating sequential models are limited since these sequential models have been focusing mainly on the single domain tasks. To generate accurate CDR models, there are some essential points to be addressed. First, CDR models should not treat source domain and target domain as equally important. Almost all existing CDR models extract information from both domains in a similar manner, using the same method (e.g., MF). Second, CDR models should consider weighing important historical items and do not treat all items as equally important. Third, CDR models should learn different user's preference for different domains.

To overcome the limitations of single-domain models and to obtain different user's preferences from different domains, we propose Cross-Domain Attentive Sequential Recommendations Based on General and Current User Preferences (CD-ASR) Model. Our intention is to utilize the source domains to extract the user's general preferences, while learning the user's current preferences in the target domain, and then fuse these preferences to make the next item prediction. Transferring knowledge using CDR can be centered based on overlapped users, overlapped items, both, or none. This chapter concentrates on different (but related) domains that share users while the items are different (e.g., Movies, Musics). Naturally, users interact similarly with different items from different domains. This assumption indicates that it is beneficial to transfer knowledge based on overlapped users across domains.

The contribution of this chapter can be summmarized as follow:

1. We propose a Cross-Domain Attentive Sequential Recommendations Based on General and Current User Preferences model (CD-ASR) to alleviate the sparsity issue of a single domain. To the best of our knowledge, this is the first cross-domain model for sequential recommendations utilizing the attention mechanism.
2. We extract user information from the source and target domains based on general and current preferences, respectively, by joining different task models
3. We perform extensive experiments using three real-world cross-domain datasets to

investigate the effectiveness of applying CDR on sequential recommendation using the self-attention mechanism. Furthermore, we compare our model with state-of-the-art sequential recommendation models.

The remaining of the chapter is organized as follows: we review single-domain and cross-domain Recommender Systems related work in Section 6.2. In Section 6.3, we formally describe the problem of CDR. Our model is proposed in Section 6.4. Section 6.5 describes the experimental setup. We discuss the results in Section 6.6, and conclude the paper in Section 6.7.

6.2 Related work

In this section, we highlight the state-of-the-art methods for non-sequential and sequential Recommender Systems in terms of a single domain. Also, we review existing cross-domain Recommender System models and point out the need for cross-domain models targeting user's sequence of items.

6.2.1 Non-sequential recommendations

Non-sequential Recommender System models usually focus on extracting users' general representational vectors for prediction. Matrix Factorization (MF) (Hu et al., 2008; Lee et al., 2013; Wang et al., 2016; Zhong et al., 2019) is a well-known model-based Collaborative Filtering (CF) approach that treats user-item interactions as a set of pairs. MF (Hu et al., 2008; Lee et al., 2013) factorizes user-item interactions matrix into low dimensional vectors and multiply these vectors for either rating or ranking prediction. Neighborhood based model is a memory-based CF approach that recommends based on calculating user similarity (Wang et al., 2017) or item similarity (Wang et al., 2006). Later on, some item similarity models (e.g., FISM (Kabbur et al., 2013)) represent a user vector by aggregating the representational vectors of the items the users have interacted with. However, these models treat all interaction items as equally important. To improve the above models, some Deep

Learning methods have been introduced. Neural Collaborative Filtering (NCF) (He et al., 2017b) combine MF with Multi-Layer Perceptron (MLP) to capture linear and non-linear characteristics of user-items interactions. Other models have adopted auto-encoder (AE) networks to reduce the dimensionality, while still obtaining good general representations of users and items (Li and She, 2017; Strub et al., 2016). In term of item similarity, NAIS (He et al., 2018) introduced attention layers into FISM to help the model concentrate more on the important items in the user historical record. These models focus only on users' general preference for prediction and do not consider the order of user-item interactions.

6.2.2 Sequential recommendations

Sequential Recommender System models predict the user's next decision by observing (treating) her historical behaviors as a sequence of actions. Earliest sequential Recommender System models use Markov Chains (MC) to predict the next action based on the last interacted item, (Zimdars et al., 2013) thus modeling first-order item transition, or alternatively, predict the next action based on the last few interacted items (He and McAuley, 2016a; He et al., 2016a), thus modeling high-order item transition. Obviously, Markov Chain based models cannot obtain users' dynamic preferences by only considering the last few recent items. Backed by Deep Learning, sequential Recommender System models were able to consider a longer sequences. The first deep learning sequential models utilize recurrent neural networks (RNN) (Donkers et al., 2017; Hidasi and Karatzoglou, 2018; Hidasi et al., 2015; Li et al., 2017; Quadrana et al., 2017; Tan et al., 2016; Yu et al., 2016) to capture short and long-term dependencies between items in the sequence. For session-based recommendation using RNN, GRU4Rec (Hidasi et al., 2015) was the first model to use Gated Recurrent Units (GRU) to distill information based on sessions. Hidasi and Karatzoglou (2018) proposed a top-N recommendation model based on GRU4Rec. Also, CNN based models (Tang and Wang, 2018; Yuan et al., 2019) use different filter sizes to extract a user's short-term preferences. Lately, attention layers have been introduced to sequential recommendation to solve the issue of vanishing gradients and to allow the use of long item sequences. SASRec (Kang

and McAuley, 2018) stacks multiple self-attention layers to obtain users' long-and-short term preferences. Most previous single domain focus on either obtaining general preference or dynamic preferences. To be more inclusive, CAR and FISSA utilize the attention mechanism to extract both users' static and dynamic preferences by coupling two user-preference models. However, when considering cold users (e.g., users that have very few historical records), the above models cannot extract meaningful information about users, simply because they are designed for single-domain tasks and they do not consider users' preferences from other domains.

6.2.3 Cross-domain recommendations

To solve the sparsity issue of user-item interactions, a number of studies have been conducted toward transferring knowledge across domains. In the early CDR models, MF-based methods have been dominant in the field despite some other existing neighborhood-based and clustering models. Collective MF (CMF) (Singh and Gordon, 2008) factorizes two matrices simultaneously and utilizes user-latent vectors as a bridge to transfer knowledge. Also, Li et al. (2009a) gather similar users or similar items into clusters that take similar representations and are then used for transferring knowledge - this is referred to as a code-book transfer approach. Adaptive code-book transfer learning (ACTL) (He et al., 2019) improves the original approach by Li et al. (2009a) in terms of accuracy and computation efficiency. Pan et al. (2011) extended the CMF approach to include multiple latent vector representations for the bridge users. Using a linear function like MF, these models cannot learn deep characteristics of the user-item interactions.

In term of DL, most existing CDR models have been built to extract a user's general preferences by focusing on using non-sequential Recommender System approaches. Also, they generally use the same method (e.g., MLP) to extract information from both source and target domains. For example, CoNet (Hu et al., 2018) joins two MLP models using cross-stitch networks (Misra et al., 2016) in order to transfer knowledge among domains. Recently, the auto-encoder framework with an attention mechanism (AAM) (Zhong et al.,

2020) used an auto-encoder followed by MLP to learn users' and items' latent vectors. Then, it fused the user's extracted vectors from different domains using attention layers. However, these models can only obtain user's general preference because they focus on non-sequential models. In other words, these approaches did not consider the order of user-item interactions.

In this chapter, we address the CDR problem using a sequential Recommender System model. Moreover, we aim at diversifying the source and target models, instead of using the same approach for both. We consider the information learned from the source domain as representing general preferences, while the information learned from the target domain (where prediction should happen) as users' current preferences.

6.3 Problem description

Our model aims at solving the problem of cross-domain sequential recommendation, considering the scenario where source-target users overlap, as this scenario is consistent with the real world situation, where a user engages with multiple domains. In our case, we have a *source* domain denoted as S and a *target* domain denoted as T. We use U to denote the set of overlapping users. Furthermore, we use I_S and I_T to denote the sets of source items and target items, respectively. Each $u \in U$ has engaged with some m source items and some n target items. Thus, user u has a sequence of source items denoted as S_u and a sequence of target items denoted as T_u which the user has interacted with. Formally, we have: $S_u = [s_{i_1}, s_{i_2}, \cdots, s_{i_m}]$, $T_u = [t_{i_1}, t_{i_2}, \cdots, t_{i_n}]$ as the source item sequence and the target item sequence for user u, sorted in a temporal order (from old to new). Our task is to transfer the user's general preferences from the source domain to accurately predict the next item t_{i_n+1} in the target domain.

Thus, our intention is to use the model specific to the source domain to extract general information about the preferences of a user, while a model specific to the target will extract preferences in the target domain. Together, the general information extracted from the source, together with the domain-specific information from the target, can presumably result in a more accurate cross-domain recommendations, as compared to appropriate baselines.

6.4 Proposed model

In this section, we describe the model proposed to address the CDR problem defined in the previous section.

6.4.1 Source (general) representation

In order to conduct a suitable transfer knowledge approach for CDR, the user information being extracted from the source domains should not be specific, but rather general. Thus, in terms of a source domain, we aim at obtaining user's general representational vector that could help in transferring meaningful information into the target domain, instead of being a source of noise. This transferred information is meant to alleviate the sparsity issue and increase target domain accuracy. To describe the source model, we first introduce the scaled dot-product (Vaswani et al., 2017).

According to Vaswani et al. (2017), the scaled dot-product attention is defined as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$$

where Q, K, V represent the query, key, and value representations of each item, respectively. The inner product between the query and key representations is scaled by $\sqrt{d}$ to avoid large values, potentially resulting from the high dimensionality.

For the source domain, we adopt a model similar to the models in (He et al., 2020; Lin et al., 2020). The model uses attention layers to determine weights for all items in the sequence and outputs the general user vector based on aggregating all these item vectors. Hence, this model takes the source item sequence as an input. First, we fix the length of the source item sequence of all users to be equal (e.g., m=50). If users have a larger number of items, we take the most recent m items. Otherwise, padding with 0 is performed at the beginning of the sequence. Let $L \in \mathbb{R}^{|I_S| \times d}$ be the learnable source item embedding matrix, where d is the latent embedding dimensionality. Now, the input sequence can be embedded as $l = [e_{s_1}, e_{s_2}, \ldots, e_{s_m}] \in \mathbb{R}^{m \times d}$. Also, attention models for sequential data consider the order

in which events happen to capture the influence of position or time. We adopt a learnable position embedding matrix denoted as $P = [p_1, p_2, \ldots, p_m] \in \mathbb{R}^{m \times d}$. We add the input sequence item embeddings and position embeddings together to formulate the embedding layer as follow:

$$E_u^s = [e_{s_i} + p_i], i \in \{1, 2, \ldots, m\}$$

One way to obtain a user general representation is to take the weighted average of all items in the sequence. Instead, we follow the idea introduced by Lin et al. (2020) by using a learnable shared query vector $q^g \in \mathbb{R}^{1 \times d}$ to distinguish the most important items in the sequence. For simplicity, we simplify the E_u^s notation as simply E. Then, the user general preferences in the source domain can be obtain using the following equation:

$$o^s = softmax(\ q^g (EW^K)^T)\ EW^V \tag{6.1}$$

where W^K and $W^V \in \mathbb{R}^{d \times d}$ are projection matrices for keys and values, respectively. Also, o_s is the learned user's general vector from the source domain. It is worth noting that the query q^g does not embed any personal or contextual information. Luong et al. (2015) refer to this type of attention layer as location based attention. So, the output of the source model is a vector representing a user based on his or her item sequence. Thus, we utilize this representational vector to transfer knowledge into the target model.

6.4.2 Target (current) representation

Our goal is to predict the next item the user will interact with in the target domain. Therefore, our intention here is to learn the user's current (dynamic) preferences. As mentioned above, sequential recommendation using self-attention layers are capable of capturing user long-term and short-term preferences (Kang and McAuley, 2018). We find this to be a good reason to adopt a sequential model for our target domain, where we want to learn the user's current interest at this present time. To start with, we assume that a user u has interacted

with some target items $T_u = [t_{i_1}, t_{i_2}, \cdots, t_{i_n}]$.

Let $K \in \mathbb{R}^{|I_T| \times d}$ be the learnable target item embedding matrix, where d is the latent embedding dimensionality. The embedding of an input sequence is $k = [e_{t_1}, e_{r_2}, \ldots, e_{t_n}] \in \mathbb{R}^{n \times d}$. Also, the learnable position embedding matrix is $P = [p_1, p_2, \ldots, p_n] \in \mathbb{R}^{n \times d}$. Similar to the process used for the source embedding layer, we add the input sequence embeddings and position embeddings together to formulate the embedding layer as follow:

$$E_u^t = [e_{t_i} + p_i], i \in \{1, 2, \ldots, n\}$$

Following Kang and McAuley (2018), we feed the embedding layer E_u^t into a stack of hierarchical self attention layers. Thus, the self-attention layer can be seen as follow:

$$t_{att}^{(b)} = Attention(\hat{E}_u^t W^Q, \hat{E}_u^t W^K, \hat{E}_u^t W^V) \tag{6.2}$$

where $W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$ are learnable projection matrices to map the embedding matrix E_u^t into three matrices (Query, Key, value); and b denotes the number of self-attention blocks. Note that, in the sequential recommendation model, the model aggregates information from the previous n embedded items with adaptive weights to predict the next item $n+1$. Also, this hierarchy of layers assist the model to obtain user long and short-term preferences.

To endow the model with non-linearity, the output of self-attention t_{att} is then fed into a fully connected layer. This helps the model to capture complex features of item transition. The computation in the feed-forward layer can be described as follow:

$$o^t = FFL(t_{att}) = \phi(t_{att} W^{(1)} + b^{(1)}) W^{(2)} + b^{(2)} \tag{6.3}$$

where $W^{(1)}, W^{(2)}$ are $d \times d$ matrices, $b^{(2)}, b^{(2)}$ are $d-$dimensional bias vectors, and ϕ is a ReLU activation function. We utilize the same normalization and dropout layers that used in (Vaswani et al., 2017). From this model, we take the last output o_n^t at the timestamp n.

The overall CDR model is shown in the Figure 6.1. Specifically, the target model is shown on the right side, while the source models (assuming that more than one source domains

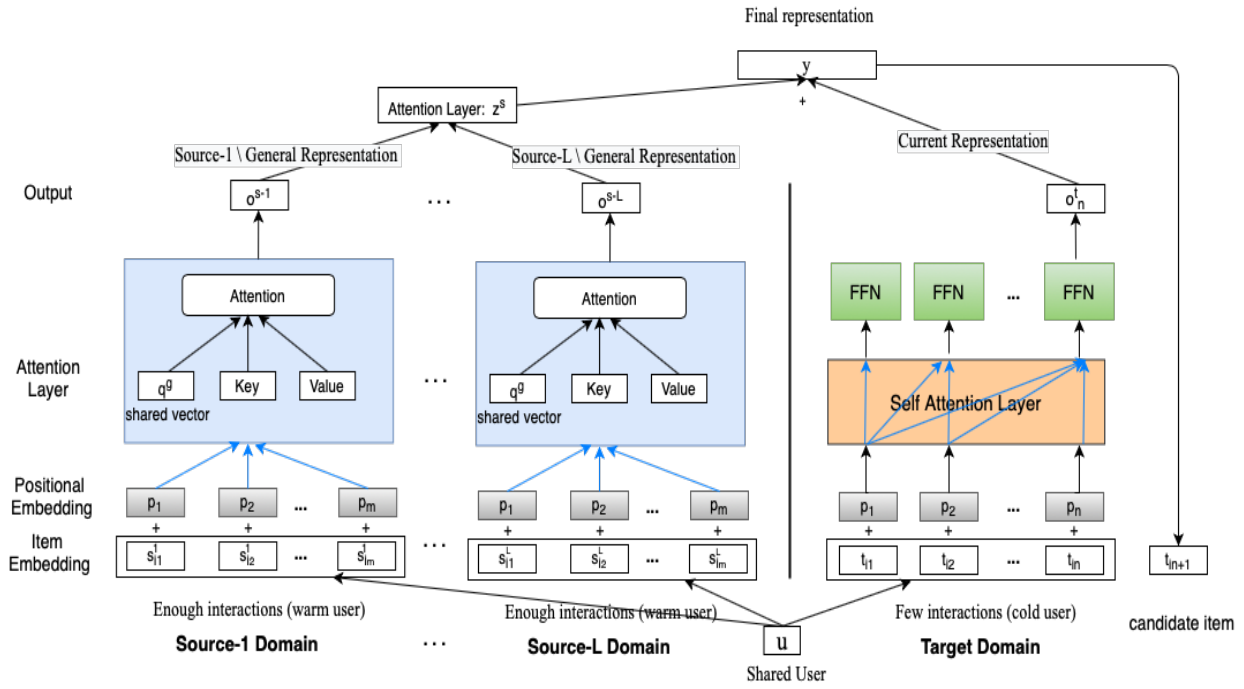


Figure 6.1: Cross-domain attentive sequential recommendations based on general and current user preferences. The source models (assuming that more than one source domains may be available) are shown on the left side. The target model is shown on the right side. The source models identify general user preferences, while the target model is focused on the specific information on the target. The outputs of the source and target models together are fed to a fully connected layer, which makes a prediction for the next item in the target sequence.

may be available) are shown on the left side.

6.4.3 Prediction and model learning

At this point, we have obtained o^s, the user general preference vector learned from the source domain, and o_n^t, the user's current preference vector learned from the target domain. Because the target domain is assumed to be very sparse, in the sense that users have very few items to learn from, we use both learned vectors together as the final representation $y^u = [o^s + o_n^t]$. In the prediction layer, we utilize MF to calculate the relevance score of the the candidate items t_{i_f} being the next predicted item for user u as: $r_{i,f}^u = \sigma(y^u \cdot (t_{i_f})^T)$. Ideally, our model should rank the next target item $t_{i_{n+1}}$ as high as possible in the list (at the top of the list).

The loss function that our model optimizes (using the Adam optimizer) is the binary cross-entropy loss defined as:

$$L = \sum_{u \in U} \sum_{f=[1,2,...,n]} -[\log(r_{i,f}) + \log(1 - r_{j,f})]$$

where $j \in I_T \setminus T_u$ is a randomly sampled negative item for each prediction.

6.5 Experimental setup

6.5.1 Dataset

We choose the existing Amazon dataset (He and McAuley, 2016b) because it is a multi-domain dataset that has partial overlap between users across different domains. Specifically, we select the largest three categories Books, Movies, and Music, as three different but related domains. We extract our datasets by considering three-domain subsets (e.g., two source domains and one target domain) and as well as two-domain subsets (e.g., one source and one target). We filter out users with less than ten interactions. Our task is to alleviate the sparsity of user-item interactions in the target domain (e.g., Music) by transferring information from one or more source domains (e.g., Books, and/or Movies). Thus, the goal is to improve the performance of the target domain in term of overall accuracy. Dataset statistics are summarized in Table 6.1 based on different pairs of source-target domains, and in Table 6.2 based on different triplets of source-source-target domains.

Table 6.1: Statistics on the **source→target** pairs of Amazon domains.

Source→Target pair	**Books → Music**		**Movies → Music**	
# users	4,427		5,465	
Domain	**Books**	**Music**	**Movies**	**Music**
# items	122,477	50,012	38,091	53,355
# interactions	308,525	201,315	310,672	268,628

Following (He and McAuley, 2016a; He et al., 2017a; Kang and McAuley, 2018; Rendle et al., 2012), we treat the available user-item interactions as implicit feedback by converting users' actual rating into binary values. We use the timestamp to determine the order of the

Table 6.2: Statistics on the **source→target** pairs of Amazon domains.

Source,**Source→Target** pair	**Books, Movies → Music**			**Books, Music → Movies**		
# users	2,440			2,440		
Domain	**Books**	**Movies**	**Music**	**Books**	**Music**	**Movies**
# items	104,927	34,092	44,273	104,927	44,273	34,092
# interactions	234,637	209,632	147,926	234,637	147,926	209,632

item sequence. Furthermore, our data was split into training and testing sets based on the standard *leave-one-out* strategy. Thus, the last item is kept for testing, and the second-to-last is used for validation while the remaining items are utilized to train the model.

To make sure that our model tackles the data sparsity problem extensively, we consider three different scenarios for user interactions in the target domain. First scenario is set to include all existing user interactions in the target domain. The second scenario is set to include only the first 10 interactions of the user in the target domain. To make the target domain even more sparser and the task for our model more challenging, the third scenario is set to include only the first 5 interactions for the user in the target domain. In this last case, having only 5 interactions, three items only are used for training and one item is used for validation, while the last item is used for testing. In each scenario, however, we utilize all user interactions in the source domain(s) to evaluate the efficiency of our model in transferring knowledge across domain.

6.5.2 Baselines

We evaluated the **CD-ASR** model against several state-of-the-art sequential recommendation models:

- **Pop:** This model ranks items based on their popularity (i.e., total number of interactions associated with each item), and thus it is a non-personalized model.
- **Bayesian Personalized Ranking (BPR-MF):** This is a general personalized recommender system using a simple ranking method for implicit feedback, combined with

MF (Rendle et al., 2012).

- **Factorized Personalized Markov Chains (FPMC):** This is a sequential model using factorization method to model first order Markov Chains, and combine them with MF (Rendle et al., 2010).

- **Self-Attentive Sequential Recommendation (SASRec):** This is a state-of-the-art sequential recommendation model (Kang and McAuley, 2018) that applies a series of self-attention and fully connected layers over the item sequence to capture the user's dynamic preferences.

- **Fusing Item Similarity Models with Self-Attention Networks for Sequential Recommendation (FISSA-lg):** This is a single domain sequential model that models user's general and local preferences and fuses them together to make predictions.

Note that some existing sequential models, for example, GRU4Rec (Hidasi et al., 2015) and Caser(Tang and Wang, 2018), were not included as baselines, as Kang and McAuley (2018) have already shown that the SASRec model outperforms these types of single models significantly.

Also, we did not include existing cross-domain models as they are mainly non-sequential, while our model is designed for sequential tasks. A recent and the only cross-domain sequential recommendation model was proposed by Ma et al. (2019). This model addresses shared account users (e.g., a family who shares an account). Because this model cannot be directly compared with our approach, we did not include it in our baselines.

6.5.3 Evaluation metrics

We evaluate our model recommendation performance via two popular ranking metrics. The first one is Hit Ratio at 10 (HR@10) that refers to the ratio of test items that are present in the top-10 recommended items. The second metric is Normalized Discounted Cumulative Gain at 10 (NDCG@10) and focuses on the exact position of the item in the top-10. Thus, it gives a higher score if the item is ranked higher. We follow the approach used by He

et al. (2017b) and (Koren, 2008) to reduce the computation cost. Thus, for each test item, we randomly sampled 99 non-interacted items for each user. Then, we evaluate our model ability to rank the test item higher in the list of 100 items.

6.5.4 Model setting

We use TensorFlow to implement the BMR-MF and FPMC models. For SASRec and FISSA-lg, we utilized the code provided by the authors. Our implementation of CAS is adopted from the published codes of SASRec and FISSA, with some modifications to the attention module, and the addition of an extra attention layer. The latent vector size is set to 50, except for Pop model. We utilize Adam (Kingma and Ba, 2014) to optimize all the baselines. The validation set is used to tune the hyper-parameters of the models.

Our model follows the architecture proposed by Kang and McAuley (2018) for the target domain. We set the learning to 0.001, while the batch size is set to 128. To prevent over-fitting, the dropout strategy is utilized, and fine-tuned using the following values/rates $0.5, 0.6, 0.7, 0.8, 0.9$. We found that the best rate for our models/data is 0.5 for both the source and target domains. The sequence length is set to 50 for the source domain, and it depends on the scenario employed for the target domain (i.e., 50, 10, or 5 items, respectively). If the sequence is less than the specified length, we perform padding at the beginning of the sequence.

6.6 Results and discussion

In this section, we present the results of our cross-domain model (CD-ASR) compared with other baselines in sequential recommendation. Also, we show the overall performance of the model. During our experiment, we investigate the effectiveness of our model to transfer knowledge from one source domain or two source domains, respectively, into the target domain. Furthermore, we adopt three different scenarios in terms of the number of user-interactions in the target domain, denoted as: *all*, only 10, and only 5 items available for the

user. A smaller number of interactions makes the target domain sparser and the task for our cross-domain model more difficult. First, we discuss transferring knowledge from one source domain into the target domain using the Book→Music and Movie→Music pairs. Then, we examine our model's efficiency with two source domains as compared to one source domain, specifically, Books, Movies→Music, and Books,Music→Movies.

6.6.1 Transfer from one source domain and one target domain

The results for the experiments with one source domain and one target domain are shown in Tables 6.3 and 6.4 for the pairs Book→Music and Movies→Music, respectively. For the Books→Music pair, our model significantly surpasses all the baseline models for all three scenarios, as shown in Table 6.3. Compared with SASRec and FISSA-lg, our model's performance is higher, especially as we consider sparser target scenarios, such as only 10 or only 5 items available for the target users.

For the Movies→Music pair, as shown in Table 6.4, our model continues to produce significantly better results as compared to the baselines. Remarkably, with only 5 item scenario, our model's results improved by 16% and 19% in comparison with the results of FISSA-lg in both HR@10 and NDCG@10, respectively. This clearly indicates the robustness of our transfer knowledge method for sequential recommendations.

Together, these results show the effectiveness of our cross-domain model and its ability to transfer knowledge. Furthermore, the results show that the single-domain models suffer more when there are only few user-item interactions in the target domain of interest.

6.6.2 Transfer from two source domains and one target domain

We also investigate the ability of our cross-domain model (CD-ASR) to aggregate user source information from two separate domains considering the three different scenarios of sparsity in the target domain. In the first case, we choose Books and Movies as source domains and transfer knowledge to enhance recommendation in the Music domain (target). Also, the results are compared with transferring knowledge from Book-only and Movie-only as shown

Table 6.3: Performance results for the Books→Music pair

Source → target	Books → Music					
scenario	all		10		5	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Pop	0.2974	0.1443	0.1778	0.0921	0.1195	0.0603
BPR-MF	0.2821	0.1644	0.1965	0.1088	0.152	0.0759
FPMC	0.3566	0.2171	0.2062	0.1131	0.1633	0.0991
SASRec	0.4276	0.271	0.2573	0.1392	0.2306	0.1135
CD-SASRec	0.423	0.2685	0.2697	0.1506	0.2365	0.1202
FISSA-lg	0.5238	0.3065	0.3777	0.1866	0.2543	0.1119
CD-ASR	**0.5594**	**0.3263**	**0.4772**	**0.2407**	**0.3562**	**0.1633**

Table 6.4: Performance results for the Movies→Music pair

Source → target	Movies → Music					
scenario	all		10		5	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Pop	0.3211	0.1835	0.1605	0.0942	0.1347	0.0732
BPR-MF	0.3392	0.2019	0.2177	0.1241	0.1711	0.0976
FPMC	0.4274	0.2597	0.2207	0.1282	0.1881	0.1015
SASRec	0.5074	0.3268	0.2887	0.1588	0.2699	0.1372
CD-SASRec	0.5048	0.3181	0.3107	0.1753	0.2765	0.1485
FISSA-lg	0.5604	0.3345	0.4065	0.2172	0.2801	0.1308
CD-ASR	**0.6109**	**0.3733**	**0.4889**	**0.2679**	**0.3344**	**0.1619**

in Table 6.5. Similarly, we choose Books and Music as the source domains and Movies as the target domain, as shown in Table 6.6. Next, we discuss the results for each case.

For the Books,Movies→Music pair, our model performs better in all three scenarios when using two source domain (Books and Movies) as compared with only one, Book or Movie, domain separately, as shown in Table 6.5. For example, in the scenario with only 5 items, the combination of Books and Movies as source domains has resulted in an increase by 4% and 5% compared to Book-only in HR@10 and NDCG@10, respectively.

Table 6.5: Performance results for the Books,Movies →Music pair

Source, Source → target	Books, Movies → Music					
Scenario	all		10		5	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Books, Movies → Music	**0.5716**	**0.3211**	**0.5303**	0.2516	**0.3893**	**0.1796**
Books → Music	0.5592	0.3175	0.5255	**0.2544**	0.3729	0.1710
Movies → Music	0.5516	0.2991	0.5233	0.2471	0.3844	0.1746

In the results for the Booke,Music→Movies pair shown in Table 6.6, the combination of two source domains (Books and Music) show better results with only 5 or 10 items are available for the user in the target domain. When including all user interactions in the target domain, the results of this combination (Books and Movie) show slightly worse results compared to using only the Music domain as source, and similar results to those of Book-only. This may be due to the fact that there may be enough information about the user already in the target domain and transferring more information may become a source of noise.

Table 6.6: Performance results for the Books,Music →Movies pair

Source,Source → target	Books, Music → Movies					
Scenario	all		10		5	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
Book, Music → Movie	0.5348	0.3035	**0.4413**	**0.2205**	**0.3135**	**0.1533**
Book → Movie	0.534	0.2988	0.4327	0.2155	0.3061	0.1409
Music → Movie	**0.5372**	**0.3074**	0.4311	0.2137	0.3125	0.1508

6.6.3 Ablation study

In the section, we further investigate the contribution of different number of blocks in the target domain. The target model is based on self-attention sequential recommendation that is implemented by stacking several blocks. Each block consists of an attention layer and a feed-forward fully connected layer. We utilize the Movies→Music pair to show the results of one, two, and three blocks as shown in Table 6.7. Stacking two or three blocks (a much deeper model) shows a significant improvement compared to when only one block is used for the *all* and only 10 scenarios. However, when using only 5 items, there is a slight improvement for the one-block architecture model. The reason may be that when there is a small number of interactions, a deeper model may not show a significant improvement as there may not be enough data to train it well (a deeper model has a larger number of parameters and thus it requires more data to train).

Table 6.7: Variation of performance with the number of blocks in the target model, when experimenting with the Movies→Music pair

Source → target	Movie → Music					
scenario	all		10		5	
Metrics	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
CD-ASR-b1	0.5802	0.3577	0.4635	0.2455	0.3309	0.1546
CD-ASR-b2	0.6109	**0.3733**	0.4869	0.2619	0.3344	0.1619
CD-ASR-b3	**0.6111**	0.3723	**0.4894**	**0.2674**	**0.3412**	**0.1682**

6.7 Summary

In this chapter, we proposed a novel model, Cross-Domain Attentive Sequential Recommendations (CD-ASR) Based on General and Current User Preferences to combine general information about a user's preferences, extracted from multiple source domains, with specific information extracted from a target domain to improve recommendation accuracy in the target domain. We extracted user general information from the source domains using

a general attention model. Simultaneously, we obtain user dynamic preferences in the target domain using a self-attention sequential model. We fused these pieces of information together to alleviate the sparsity problem in the target domain. We considered three different interaction scenarios in the target domain to investigate the robustness of our model against the sparsity issue. Using two popular metrics, we evaluated our model using three large datasets considering one and two source domains to transfer knowledge into the target model. Experimental results show that our cross-domain model significantly outperforms all the baselines considered.

Chapter 7

Conclusions and future work

In this dissertation, I have studied cross-domain recommender systems with the goal of reducing the sparsity problem of a single domain, and increasing the recommendation accuracy. Using deep neural networks, I proposed one non-sequential cross-domain recommender system model and two sequential models. The main contributions of this dissertation are summarized as follows.

In Chapter 4, I proposed a cross-domain recommender system based on the neural collaborative filtering model. The model utilizes linear and non-linear sub-networks to learn the embedding vectors of users and items in the source domain. Then, these user and item embedding vectors learned from the source domain are used to transfer knowledge into the target domain. Specifically, we sum up all source item embedding vectors (the items that the current user u has interacted with in the source domain) at the embedding layer of the target model. Then, all the embedding vectors are concatenated and fed into a deep neural network. The Amazon dataset with its various domains was utilized to evaluate our proposed cross-domain model. Experimental results have shown that transferring knowledge across domain has led to an increase in the recommendation accuracy.

In Chapter 5, I proposed a cross-domain self-attentive sequential recommendations model. In the target domain, the model encodes the user target item sequence to predict the next item that the user will most likely interact with. The user source representational vec-

tor was obtained by applying the state-of-the-art self-attention sequential recommendation model over the user source item sequence. We utilized the early fusion method to transfer the extracted knowledge (the pre-learned user source representational vector) into the target domain to increase the recommendation accuracy. Several source-target pairs extracted from the widely used Amazon dataset are used to evaluate the effectiveness of our proposed cross-domain model, while experimenting with different levels of sparseness over the user interactions. This is simulated by assuming that only 5 or only 10 items are available for the user in the target domain. Experimental results show an improvement in the recommendation accuracy in the target domain when some knowledge has been transferred form the source domain.

In Chapter 6, I proposed a cross-domain attentive sequential recommendations using general and current user preferences. In this cross-domain approach, we transfer knowledge form one or more source domains into the target domain. For the source domains, we apply general models over the user source item sequences to extract user general (source) representational vectors for each source domain. For the target domain, we adopt the self-attentive sequential recommendation to obtain user dynamic preferences as a user target representational vector. At the end, we combine these user general vectors with the user target vector together to predict the next item for the user in the target domain. Experimental results on several pairs of Amazon's largest and most related domains have shown that our model significantly outperforms the baselines in terms of recommendation accuracy.

Because sequential recommendation systems model item sequences, our future work will focus on introducing a user latent (embedding) vector, in addition to the learned item sequence vector, to improve the recommendation accuracy even further. Also, self-attentive sequential recommendation systems learn to track the order of items by using an order embedding (e.g., 1,2,...,50). In our future work, we will introduce a real time embedding (e.g., using timestamp for each user-item interaction) that will not only keep track of the sequence order, but also embed useful information that indicates how long it takes between two interactions. In other words, if two user-item interactions are apart by one day or by one week, this additional information may contributes differently into the recommender systems.

Lastly, when using a larger number of source domains (e.g., 4 or 5 source domains) are used to transfer knowledge into the target domain, this increases the chance that one of these source domains may contain noise information that could degrade the recommendation performance in the target domain. One of our future work ideas is to investigate this issue by, for example, applying a special attention layer that could eliminate the source domain with high noise information.

Bibliography

T. Adler, J. Brandstetter, M. Widrich, A. Mayr, D. Kreil, M. Kopp, G. Klambauer, and S. Hochreiter. Cross-domain few-shot learning by representation fusion. *arXiv preprint arXiv:2010.06498*, 2020.

C. C. Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

N. Alharbi and D. Caragea. Cross-domain recommender system based on neural collaborative filtering. In *Proceedings of the 17th International Conference on Machine Learning and Data Mining (MLDM 2021)*, 2021a.

N. Alharbi and D. Caragea. Cross-domain self-attentive sequential recommendations. In *Proceedings of the 2nd International Conference on Data Science and Applications (ICDSA 2021)*, 2021b.

N. Alharbi and D. Caragea. Cross-domain attentive sequential recommendations based on general and target-specific user preferences. In *To be submitted to the 15th ACM Conference on Recommender Systems (RecSys 2021)*, 2021c.

T. Bai, J.-R. Wen, J. Zhang, and W. X. Zhao. A neural collaborative filtering model with interaction-based neighborhood. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1979–1982. ACM, 2017.

I. Bayer, X. He, B. Kanagal, and S. Rendle. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1341–1350, 2017.

J. Bennett, S. Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. Citeseer, 2007.

I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi. Cross-domain recommender systems. In *Recommender Systems Handbook*, pages 919–959. Springer, 2015.

H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

K.-Y. Chiang, C.-J. Hsieh, and I. S. Dhillon. Matrix completion with noisy side information. In *Advances in Neural Information Processing Systems*, pages 3447–3455, 2015.

M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. *Recommender systems handbook*, pages 107–144, 2011.

T. Donkers, B. Loepp, and J. Ziegler. Sequential user-based recurrent neural network recommendations. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 152–160, 2017.

G. K. Dziugaite and D. M. Roy. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*, 2015.

H. Fang, D. Zhang, Y. Shu, and G. Guo. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *arXiv:1905.01997*, 2019.

S. Gao, H. Luo, D. Chen, S. Li, P. Gallinari, and J. Guo. Cross-domain recommendation via cluster-level latent factor model. In *ECML*. Springer, 2013.

M. A. Ghazanfar and A. Prugel-Bennett. A scalable, accurate hybrid recommender system. In *2010 Third International Conference on Knowledge Discovery and Data Mining*, pages 94–98. IEEE, 2010.

A. Gogna and A. Majumdar. A comprehensive recommender system model: Improving accuracy for both warm and cold start users. *IEEE Access*, 3:2803–2813, 2015.

G. Hadash, O. S. Shalom, and R. Osadchy. Rank and rate: multi-task learning for recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 451–454. ACM, 2018.

M. He, J. Zhang, and S. Zhang. Actl: Adaptive codebook transfer learning for cross-domain recommendation. *IEEE Access*, 7:19539–19549, 2019.

R. He and J. McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*, pages 191–200. IEEE, 2016a.

R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517, 2016b.

R. He, C. Fang, Z. Wang, and J. McAuley. Vista: A visually, socially, and temporally-aware model for artistic recommendation. In *RecSys*, 2016a.

R. He, W.-C. Kang, and J. McAuley. Translation-based recommendation. In *RecSys*, pages 161–169, 2017a.

X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 549–558, 2016b.

X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017b.

X. He, Z. He, J. Song, Z. Liu, Y.-G. Jiang, and T.-S. Chua. Nais: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2018.

Y. He, Y. Zhang, W. Liu, and J. Caverlee. Consistency-aware recommendation for user-generated item list continuation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 250–258, 2020.

B. Hidasi and A. Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*, pages 843–852, 2018.

B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

C.-K. Hsieh, L. Yang, Y. Cui, T.-Y. Lin, S. Belongie, and D. Estrin. Collaborative metric learning. In *Proceedings of the 26th international conference on world wide web*, pages 193–201. International World Wide Web Conferences Steering Committee, 2017.

G. Hu, Y. Zhang, and Q. Yang. Conet: Collaborative cross networks for cross-domain recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 667–676, 2018.

L. Hu, S. Jian, L. Cao, Z. Gu, Q. Chen, and A. Amirbekyan. Hers: Modeling influential contexts with heterogeneous relations for sparse and cold-start recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3830–3837, 2019.

Q.-Y. Hu, Z.-L. Zhao, C.-D. Wang, and J.-H. Lai. An item orientated recommendation algorithm from the multi-view perspective. *Neurocomputing*, 269:261–272, 2017.

Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

L. Huang, Z.-L. Zhao, C.-D. Wang, D. Huang, and H.-Y. Chao. Lscd: Low-rank and sparse cross-domain recommendation. *Neurocomputing*, 366:86–96, 2019.

S. Kabbur, X. Ning, and G. Karypis. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 659–667, 2013.

W.-C. Kang and J. McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206. IEEE, 2018.

M. M. Khan, R. Ibrahim, and I. Ghani. Cross domain recommender systems: a systematic literature review. *ACM Computing Surveys (CSUR)*, 50(3):1–34, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD'08*, pages 426–434, 2008.

Y. Koren and R. Bell. Advances in collaborative filtering. *Recommender systems handbook*, pages 77–118, 2015.

Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, pages 1378–1387. PMLR, 2016.

V. Kumar, A. K. Pujari, S. K. Sahu, V. R. Kagita, and V. Padmanabhan. Collaborative filtering using multiple binary maximum margin matrix factorizations. *Information Sciences*, 380:1–11, 2017.

X. N. Lam, T. Vu, T. D. Le, and A. D. Duong. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 208–211, 2008.

A. S. Lampropoulos, P. S. Lampropoulou, and G. A. Tsihrintzis. A cascade-hybrid music recommender system for mobile services based on musical genre classification and personality diagnosis. *Multimedia Tools and Applications*, 59(1):241–258, 2012.

J. Lee, S. Kim, G. Lebanon, and Y. Singer. Local low-rank matrix approximation. In *International conference on machine learning*, pages 82–90. PMLR, 2013.

B. Li, Q. Yang, and X. Xue. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *IJCAI*, volume 9, pages 2052–2057, 2009a.

B. Li, Q. Yang, and X. Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *ICML*, pages 617–624, 2009b.

J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. Neural attentive session-based recommendation. In *CIKM*, pages 1419–1428, 2017.

X. Li and J. She. Collaborative variational autoencoder for recommender systems. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 305–314, 2017.

Y. Li, J. Hu, C. Zhai, and Y. Chen. Improving one-class collaborative filtering by incorporating rich user information. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 959–968, 2010.

J. Lian, F. Zhang, X. Xie, and G. Sun. Cccfnet: a content-boosted collaborative filtering neural network for cross domain recommender systems. In *Proceedings of the 26th international conference on World Wide Web companion*, pages 817–818, 2017.

J. Lin, W. Pan, and Z. Ming. Fissa: fusing item similarity models with self-attention networks

for sequential recommendation. In *Fourteenth ACM Conference on Recommender Systems*, pages 130–139, 2020.

Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang. Stamp: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1831–1839, 2018.

B. Loni, Y. Shi, M. Larson, and A. Hanjalic. Cross-domain collaborative filtering with factorization machines. In *ECIR*, pages 656–661. Springer, 2014.

P. Lops, M. De Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, pages 73–105, 2011.

Z. Lu, E. Zhong, L. Zhao, E. W. Xiang, W. Pan, and Q. Yang. Selective transfer learning for cross domain recommendation. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 641–649. SIAM, 2013.

M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

F. Lv, T. Jin, C. Yu, F. Sun, Q. Lin, K. Yang, and W. Ng. Sdm: Sequential deep matching model for online large-scale recommender system. In *CIKM*, 2019.

M. Ma, P. Ren, Y. Lin, Z. Chen, J. Ma, and M. d. Rijke. π-net: A parallel information-sharing network for shared-account cross-domain sequential recommendations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 685–694, 2019.

T. Man, H. Shen, X. Jin, and X. Cheng. Cross-domain recommendation: An embedding and mapping approach. In *IJCAI*, pages 2464–2470, 2017.

N. Mirbakhsh and C. X. Ling. Improving top-n recommendation for cold-start users via cross-domain information. *ACM TKDD*, 9(4):1–19, 2015.

I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3994–4003, 2016.

T. Nguyen and A. Takasu. Npe: neural personalized embedding for collaborative filtering. *arXiv preprint arXiv:1805.06563*, 2018.

W. Pan, N. N. Liu, E. W. Xiang, and Q. Yang. Transfer learning to predict missing ratings via heterogeneous user feedbacks. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 130–137, 2017.

S. Rendle and C. Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 273–282, 2014.

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW'10*, pages 811–820, 2010.

S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

F. Ricci, L. Rokach, and B. Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

A. K. Sahu and P. Dwivedi. Matrix factorization in cross-domain recommendations framework by shared users latent factors. *Procedia computer science*, 143:387–394, 2018.

A. K. Sahu and P. Dwivedi. User profile as a bridge in cross-domain recommender systems for sparsity reduction. *Applied Intelligence*, 49(7):2461–2481, 2019.

A. K. Sahu and P. Dwivedi. Knowledge transfer by domain-independent user latent factor for cross-domain recommender systems. *Future Generation Computer Systems*, 108:320–333, 2020.

R. Salakhutdinov, A. Mnih, and G. Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

S. Sedhain, A. K. Menon, S. Sanner, and L. Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th international conference on World Wide Web*, pages 111–112, 2015.

U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, 1995.

A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 650–658, 2008.

F. Strub, R. Gaudel, and J. Mary. Hybrid recommender system based on autoencoders. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 11–16, 2016.

X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.

F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

Y. K. Tan, X. Xu, and Y. Liu. Improved recurrent neural networks for session-based recommendations. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 17–22, 2016.

J. Tang and K. Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, pages 565–573, 2018.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

C.-D. Wang, Z.-H. Deng, J.-H. Lai, and S. Y. Philip. Serendipitous recommendation in e-commerce using innovator-based collaborative filtering. *IEEE transactions on cybernetics*, 49(7):2678–2692, 2018.

H. Wang, N. Wang, and D.-Y. Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1235–1244, 2015.

J. Wang, A. P. De Vries, and M. J. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, 2006.

K. Wang, H. Peng, Y. Jin, C. Sha, and X. Wang. Local weighted matrix factorization for top-n recommendation with implicit feedback. *Data Science and Engineering*, 1(4):252–264, 2016.

Y. Wang, J. Deng, J. Gao, and P. Zhang. A hybrid user similarity model for collaborative filtering. *Information Sciences*, 418:102–118, 2017.

Y. Wu, C. DuBois, A. X. Zheng, and M. Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the ninth ACM international conference on web search and data mining*, pages 153–162, 2016.

C. Xiong, S. Merity, and R. Socher. Dynamic memory networks for visual and textual question answering. In *International conference on machine learning*, pages 2397–2406. PMLR, 2016.

Z. Yang, R. Salakhutdinov, and W. W. Cohen. Transfer learning for sequence tagging with hierarchical recurrent networks. *arXiv preprint arXiv:1703.06345*, 2017.

H. Ying, F. Zhuang, F. Zhang, Y. Liu, G. Xu, X. Xie, H. Xiong, and J. Wu. Sequential recommender system based on hierarchical attention network. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 729–732, 2016.

F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 582–590, 2019.

S. Zhang, Y. Tay, L. Yao, and A. Sun. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414*, 2018.

S.-T. Zhong, L. Huang, C.-D. Wang, and J.-H. Lai. Constrained matrix factorization for course score prediction. In *2019 IEEE International Conference on Data Mining (ICDM)*, pages 1510–1515. IEEE, 2019.

S.-T. Zhong, L. Huang, C.-D. Wang, J.-H. Lai, and S. Y. Philip. An autoencoder framework with attention mechanism for cross-domain recommendation. *IEEE Transactions on Cybernetics*, 2020.

F. Zhu, Y. Wang, C. Chen, G. Liu, M. Orgun, and J. Wu. A deep framework for cross-domain and cross-system recommendations. In *IJCAI International Joint Conference on Artificial Intelligence*, 2018.

A. Zimdars, D. M. Chickering, and C. Meek. Using temporal data for making recommendations. *arXiv preprint arXiv:1301.2320*, 2013.