University of Zurich^UZH

Heterogeneous Information Sources for Recommender Systems

Bachelor Thesis July 12, 2016

Marco釉nternährer
of Romoos, LU, Switzerland

Student-ID: 07-118-771
marco.unternaehrer@uzh.ch

Advisor: **Bibek Paudel**

Prof. Abraham Bernstein, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.uzh.ch/ddis

Acknowledgements

I would like to express my gratitude to Prof. Abraham Bernstein for giving me the opportunity to write this thesis. In particular, I would like to thank my advisor Bibek Paudel for his great effort and patience. He has guided me well, provided me with many valuable suggestions and carefully reviewed this work. I am also very grateful for the enduring support throughout my studies of my family and friends.

Zusammenfassung

Der meist angewandte Algorithmus für Empfehlungssysteme basiert auf kollaborativem Filtern, welcher dazu nur die Bewertungen von Objekten durch Benutzer verwendet. Diese Arbeit präsentiert zwei neue Ansätze wie zusätzliche Daten in Form von Merkmalsvektoren eingesetzt werden können, um bessere Top-N Empfehlungen abzugeben. Unser erstes Modell, MPCFs-SI, basiert auf einer nichtlinearen Matrix Faktorisierung (MPCFs) und verwendet die zusätzlichen Daten um das Modell zu regularisieren. Das zweite Modell, MFNN, ist eine Kombination aus Matrix Faktorisierung und einem neuralen Netz und spielt die zusätzlichen Daten als Eingabe ins neurale Netz ein. Unsere Experimente haben gezeigt, dass MPCFs-SI bessere Leistungen erbringt als das beste Vergleichsmodell MPCFs auf unseren MovieLens 100k und MovieLens 1M Teildatensätzen. MFNN ist schlechter als das Vergleichsmodell MPCFs auf dem MovieLens 100k Teildatensatz, jedoch ist es auf einem ähnlichen Performanzlevel wie MPCFs-SI auf dem grösseren MovieLens 1M Teildatensatz.

Abstract

The most popular algorithm for recommender systems utilizes the collaborative filtering technique which makes only use of the user-item rating matrix. This thesis introduces two approaches which employ extra data encoded as feature vectors. One of our proposed models, MPCFs-SI, is based on a nonlinear matrix factorization model for collaborative filtering (MPCFs) and utilizes the extra data to regularize the model. The second model called MFNN is an ensemble of a matrix factorization and a neural network and uses the extra data as an input to the neural network. Our results show that MPCFs-SI outperforms the baseline recommender MPCFs on a subset of both MovieLens 100k and MovieLens 1M datasets. MFNN is inferior to the MPCFs model on our MovieLens 100k subset, however, it is at a similar performance level as MPCFs-SI on the bigger MovieLens 1M subset.

Table of Contents

1

Introduction

Recommender systems are software systems suggesting items to users which might fit their personal tastes. Such systems play a crucial part in e-commerce, social networks, news, music and video websites [Christoffel, 2014, Almahairi et al., 2015].

This work investigates methods that employ data of users, items, and by users on items. In this chapter, the first section gives a brief overview of different existing methods for recommending systems. The purpose of this research writing is stated in 1.2 and the chapter is concluding with a brief discussion of related work. In Chapter 2, methods incorporating extra data into recommender systems are discussed, evaluation of the proposed methods are then compared and analyzed in Chapter 3.

1.1 Background

Recommender systems are usually classified according to their approach to what kind of input data they use [Christoffel, 2014]. In general, we can distinguish between two groups: collaborative filtering (CF) based methods and content-based methods. Collaborative filtering methods make use of the user-item rating matrix in order to build models [Kabbur and Karypis, 2014]. Among all the CF algorithms, the most successful ones are the latent factor models [Bao et al., 2014]. These models try to explain user ratings by characterizing both items and users on factors inferred from the rating patterns [Bao et al., 2014]. Content-based methods on the other hand use meta data about the items in order to analyze characteristics of items that a user liked in the past and searches for similar items [Christoffel, 2014]. A combination of several methods is referred to as hybrid recommender systems [Ricci et al., 2011]. The purpose of such hybrid systems is to use the advantages of one method and fix the disadvantages of another [Ricci et al., 2011].

While CF techniques are very popular and widely used, one of its main disadvantages is known as the cold-start problem [Christoffel, 2014]. In collaborative filtering an item can only be evaluated as useful for a user once it received quality estimations from users considered as similar to the current user [Christoffel, 2014]. The same problem arises for new users of the systems with none or only very few rated or liked items [Christoffel, 2014]. The cold-start problem is a result of a challenge that most recommender systems face, namely a sparse user-item-feedback matrix [Christoffel, 2014].

When CF is casted as a problem of matrix factorization with missing values, this data sparsity easily leads to naive matrix factorization overfitting the training set of observed ratings [Almahairi et al., 2015, Ilin and Raiko, 2010]. It has been shown that better rating prediction can be obtained by regularizing the matrix factorization using an additional source of information [McAuley and Leskovec, 2013, Bao et al., 2014, Almahairi et al., 2015]

1.2 Motivation

Collaborative filter techniques suffer from the cold-start problem. In this work, our goal is to improve top-N recommendation and address the cold-start problem by incorporating side information encoded as feature vectors into recommender. Specifically, we study how feature vectors extracted from movie subtitles can be used to improve the performance of matrix factorization models.

We introduce two approaches to incorporate side information into recommender and compare these to state-of-the-art-based approaches [Kabbur, 2015, Ning and Karypis, 2011, Rendle et al., 2009].

1.3 Related Work

Recent works attempt to combine latent rating modeling with latent review topics [McAuley and Leskovec, 2013]. Their approach, called **HFT**, more accurately predicts product ratings by harnessing the information present in review texts. This is especially true for new products and users, who may have too few ratings to model their latent factors, yet may still provide substantial information from the text of even a single review [McAuley and Leskovec, 2013]. HFT uses latent Dirichlet allocation for modeling the user-review texts.

[Bao et al., 2014] also consider the rating and the accompanied review text. While in the HFT model by [McAuley and Leskovec, 2013] the latent review topics might only relate to latent user (or item) factors, the model of [Bao et al., 2014] simultaneously correlate latent topics with latent user and item factors. The presented model, called **TopicMF**, is a matrix factorization model and uses nonnegative matrix factorization as the topic modeling technique of the review texts. [Bao et al., 2014] show on 22 real-world datasets that their method handles the data sparsity better than three state-of-the-art methods.

It has been observed earlier, that matrix factorization approaches easily overfit on the rating data, leading to poor generalization performance [Almahairi et al., 2015]. [Almahairi et al., 2015] have shown that utilizing review texts as a way of regularizing the derived item representations improve the generalization performance of a rating prediction model. While previous state-of-the-art approaches have used latent Dirichlet allocation for modeling the user-review texts, [Almahairi et al., 2015] propose two neural network based models: a distributed bag-of-words **BoWLF** and a recurrent neural network approach **LMLF**.

All these approaches influenced our work. For this work, extra data from movie subtitles were used while the mentioned works use extra data of user-item pairs i.e review texts. [Bao et al., 2014] argue that if a rating score is associated with a review text, they might understand why the user liked or disliked the item and thus make better predictions. In contrast, our work is more general for several reasons. First, additional user-item data might not be available. Secondly, raw user or item data might be too sensitive to work with directly and thus already be encoded into feature vectors. And finally, the best suiting machine learning techniques for the kind of extra data can be used to extract feature vectors.

2

Incorporating Extra Data into Recommender

In this work, we focus particularly on nonlinear methods for collaborative filtering. After giving an overview of the used notation in the first section, we show a recently developed nonlinear matrix factorization technique proposed in [Kabbur, 2015] (Section 2.2), then Section 2.3 introduces two models which make use of extra data.

2.1 Notation

As in [Kabbur, 2015], all vectors are represented by bold lower case letters (e.g. $\mathbf{a}$), all matrices are represented by bold upper case letters (e.g. $\mathbf{A}$) and predicted values are denoted by having a ˆ (hat) over it (e.g. $\hat{a}$). All important symbols and definitions used in this work are summarized in Table 2.1.

2.2 Nonlinear Matrix Factorization for Collaborative Filtering

Nonlinear methods make less a priori assumptions on how components of a system may interact [Carello and Moreno, 2005]. A recently developed nonlinear method for top-N recommendation is called **MaxMF** [Weston et al., 2013]. MaxMF extends the matrix factorization based approaches by representing a user with multiple latent vectors, each corresponding to a different "taste" associated with the user [Kabbur, 2015]. These different tastes associated with each user representation are termed as *interests* [Kabbur, 2015]. The assumption behind this approach is that it helps to capture user preferences better, especially when the user's interests are diverse [Kabbur, 2015].

[Kabbur, 2015] developed a nonlinear matrix factorization method called **MPCF**, which is based on MaxMF. While MaxMF assumes that interest-specific preferences of the users are completely different, MPCF models the user as a combination of global preference and interest-specific latent factors [Kabbur, 2015]. In the work of [Kabbur, 2015], given a user u, an item i and T user local preferences, the estimated rating $\hat{r}_{ui}$ is given

Symbol	**Definition**
$u,\ i$	Individual user u, item i
$n,\ m$	Number of users and items
k	Number of latent factors
T	Number of user local preferences
r_{ui}	Rating by user u on item i
$\hat{r}_{ui}$	Predicted rating for user u on item i
$\mathbf{R}$	User-item rating matrix, $\mathbf{R} \in \mathbb{R}^{n \times m}$
$\mathbf{P}$	User latent factor matrix, $\mathbf{P} \in \mathbb{R}^{n \times k}$
$\mathbf{Q}$	Item latent factor matrix, $\mathbf{Q} \in \mathbb{R}^{m \times k}$
$\mathbf{S}$	User latent factor tensor, $\mathbf{S} \in \mathbb{R}^{n \times k \times T}$
$\mathbf{B_u}$	User-Interest bias matrix, $\mathbf{B_u} \in \mathbb{R}^{n \times T}$
$\mathbf{b_i}$	Item bias vector, $\mathbf{b_i} \in \mathbb{R}^{m}$
λ_{reg}	l_2-regularization constant
d	Number of Doc2Vec dimensions
$\mathbf{f}_i$	Doc2Vec vector for item i, $\mathbf{f}_i \in \mathbb{R}^{d}$
$\hat{\mathbf{f}}_i$	Predicted Doc2Vec vector, $\hat{\mathbf{f}}_i \in \mathbb{R}^{d}$
$\mathbf{G}$	Weight matrix of the Doc2Vec prediction model, $\mathbf{G} \in \mathbb{R}^{d \times k}$
$\mathbf{h}$	Bias vector of the Doc2Vec prediction model, $\mathbf{h} \in \mathbb{R}^{d}$
λ_{ed}	Parameter that balances matrix factorization model and the Doc2Vec prediction model
λ_{cos}	Parameter that balances the Euclidean and the cosine distance
$\mathbf{w}, \mathbf{W'}$	Weight vector and matrix of the neural network
$\mathbf{x}$	Input vector to the neural network, $\mathbf{x} \in \mathbb{R}^{2k+d}$
$\varphi(x)$	Activation function of the neural network

Table 2.1: Symbols used and definitions

by the sum of the estimations from global preference and interest-specific preference components. That is,

$$\hat{r}_{ui} = \mu + b_i + \mathbf{p}_u \mathbf{q}_i^\intercal + \max_{t=1,..,T} (b_{ut} + f(u, i, t)), \tag{2.1}$$

where μ is the global bias i.e., the average rating value of the entire data set, b_i is the item bias corresponding to item i, b_{ut} is the user-local preference bias corresponding to user u and local preference t, $\mathbf{p}_u$ is the latent vector associated with user u and $\mathbf{q}_i$ is the latent vector associated with item i [Kabbur, 2015]. [Kabbur, 2015] proposed two different methods to represent the interest-specific preference component *f(u, i, t)*. The first one, named **MPCFi**, has independent item factors in *f(u, i, t)* compared to that of global preference component, whereas the second one shares the item factors of *f(u, i, t)* with the global preference component, and thus named **MPCFs**. The local preference component *f(u, i, t)* for MPCFs is given by,

$$f(u, i, t) = \mathbf{s}_{ut} \mathbf{q}_i^\intercal, \tag{2.2}$$

where $\mathbf{s}_{ut}$ is the user latent vector for u in the local preference component corresponding to the local preference t and $\mathbf{q}_i^\intercal$ is the shared item latent vector between the global preference and the local preference components [Kabbur, 2015]. The following regularized optimization problem is minimized to learn $\mathbf{P}$, $\mathbf{Q}$, $\mathbf{S}$, $\mathbf{B_u}$ and $\mathbf{b_i}$ matrices and vectors:

$$\mathcal{L}_{rating} = \frac{1}{2} \sum_{u,i \in R} (r_{ui} - \hat{r}_{ui})^2 + \frac{\lambda_{reg}}{2} (\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2 + \|\mathbf{S}\|_F^2 + \|\mathbf{B_u}\|_F^2 + \|\mathbf{b_i}\|_2^2), \tag{2.3}$$

where r_{ui} is the ground truth value, $\hat{r}_{ui}$ is the estimated value, $\mathbf{b_i}$ is the vector corresponding to the item biases, $\mathbf{B_u}$ is the matrix corresponding to the local preference user biases and λ_{reg} is the l_2-regularization constant for latent factor matrices [Kabbur, 2015]. [Kabbur, 2015] propose to solve the optimization problem in Eq. 2.3 using a stochastic gradient descent algorithm and provide pseudocode for that.

2.3 Using Extra Data

In this section, we discuss how feature vectors were extracted from movie subtitles, then two approaches are propose to incorporate such vectors in order to improve top-N recommendations.

2.3.1 Feature Extraction from Movie Subtitles

Movie subtitles have been from a website called Opensubtitles[1]. Note that not all subtitles belong to the movies the website has claimed, however, due to the limitation of time, a proper analysis on how many selections were affected and finding solutions were not

[1]Opensubtitles - http://opensubtitles.org

carried out. Next, a pre-processing step was removing all non-alphanumerical characters which gave a natural language document per subtitle with an average of 7521 words per document. With the help of a library called NLTK[2], each document was tokenized and stop words were removed. Gensim[3], a topic modeling library, was then used to generate a feature vector per document via a deep learning algorithm called Paragraph Vectors (also known as Doc2Vec) introduced in [Le and Mikolov, 2014]. The distributed-bag-of-words variant of the Paragraph Vector algorithm and negative sampling was used. In preliminary experiments, we have tested several Doc2Vec vector dimensions d. While some work on sentiment analysis of movie reviews use 100 dimensions [Qiu, 2015] or 300 dimensions [Chapman, 2014], we have found that a vector size of 50 resulted in higher similarities between movies and their sequels. The full parameter set is listed in Appendix A.1.1. We refer the readers to the excellent notes of [Rong, 2014] and [Goldberg and Levy, 2014] to explain in detail how Word2Vec, the common name of the base algorithm of Paragraph Vectors, works.

2.3.2 MPCFs-SI: Regularizing with Extra Data

As discussed in Section 1.3, [Almahairi et al., 2015] has shown that utilizing additional data as a way of regularizing the derived item representations has improved the generalization performance of a rating prediction model. It has accomplished to reduce overfitting by simultaneously estimating the parameters of the matrix factorization to perform well on another related task [Almahairi et al., 2015]. In a similar fashion, we propose to optimize a nonlinear matrix factorization model MPCFs in Eq. 2.3 and regularize it with a model which predicts the Doc2Vec vector of an item $\mathbf{f}_i$, given the latent vector for that item $\mathbf{q}_i$. The Doc2Vec prediction is modeled as a simple affine transformation of the item latent vector, as shown in Eq.(2.4):

$$\hat{\mathbf{f}}_i = \mathbf{G}\mathbf{q}_i + \mathbf{h}, \tag{2.4}$$

where $\hat{\mathbf{f}}_i$ is the predicted Doc2Vec vector for item i, $\mathbf{G}$ and $\mathbf{h}$ are a weight matrix and a bias vector to be learned. Since we are transforming from one vector space into another, we have decided to use the squared Euclidean and cosine distances as part of the loss term. We can informally think of minimizing the Euclidean distance as preserving the magnitude of the vector, while minimizing the cosine distance helps with the vector direction. Therefore, the loss term for the Doc2Vec prediction model is given in Eq. 2.5:

$$\mathcal{L}_{ed} = \sum_{u,i \in R} (\|\mathbf{f}_i - \hat{\mathbf{f}}_i\|_F^2 + \lambda_{cos} D_C(\mathbf{f}_i, \hat{\mathbf{f}}_i)) + \lambda_{reg} \|\mathbf{G}\|_F^2, \tag{2.5}$$

where $\mathbf{f}_i$ is a Doc2Vec vector for item i extracted from movie subtitles via deep learning methods discussed in the previous section, λ_{cos} is a parameter that balances the performance of the squared Euclidean distance and the cosine distance, $D_C(\mathbf{f}_i, \hat{\mathbf{f}}_i)$ is the cosine

[2]NLTK - http://www.nltk.org/
[3]Gensim - https://radimrehurek.com/gensim/

distance and λ_{reg} is the l_2-regularization constant for the weight matrix. The cosine similarity S_C and the cosine distance D_C between two vectors **a** and **b** are defined as,

$$S_C(\mathbf{a}, \mathbf{b}) = \frac{\mathbf{a} \cdot \mathbf{b}}{\|\mathbf{a}\|\|\mathbf{b}\|} \in [-1, 1], \tag{2.6}$$

$$D_C(\mathbf{a}, \mathbf{b}) = 1 - S_C(\mathbf{a}, \mathbf{b}) \in [0, 2], \tag{2.7}$$

where $\|\mathbf{a}\|$ is the length of the vector **a**. A cosine similarity between two vectors **a** and **b** is 1, if they point exactly in the same direction in the vector space, while a value of 0 means that the vectors are orthogonal and a value of -1 means that they point in the exact opposite direction in the vector space.

We have also considered modeling the Doc2Vec prediction with a neural network. Preliminary results have shown that a neural network model does not improve our main measure AUC.

We propose to linearly combine the matrix factorization with the Doc2Vec prediction model. Therefore, the full loss term to minimize is the following:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{rating} + \lambda_{ed}\mathcal{L}_{ed} \\ &= \frac{1}{2}\sum_{u,i \in R}(r_{ui} - \hat{r}_{ui})^2 + \frac{\lambda_{reg}}{2}(\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2 + \|\mathbf{S}\|_F^2 + \|\mathbf{B_u}\|_F^2 + \|\mathbf{b_i}\|_2^2) \\ &+ \lambda_{ed}(\sum_{u,i \in R}(\|\mathbf{f}_i - \hat{\mathbf{f}}_i\|_F^2 + \lambda_{cos}D_C(\mathbf{f}_i, \hat{\mathbf{f}}_i)) + \lambda_{reg}\|\mathbf{G}\|_F^2),\end{aligned} \tag{2.8}$$

where λ_{ed} is a parameter that balances the performance of the matrix factorization model and the Doc2Vec prediction model, which was one of the parameters to be determined by running grid search.

We apply AdaGrad, a stochastic gradient descent variant, to update $\mu, \mathbf{P}, \mathbf{Q}, \mathbf{S}, \mathbf{B_u}$ and $\mathbf{b_i}$. The gradients for the matrix factorization model are directly from [Kabbur, 2015]. We have implemented the Doc2Vec prediction model in Theano and update **G** and **h** with AdaGrad as well. Theano offers automatic differentiation such that we simply add $\lambda_{ed}\frac{\partial \mathcal{L}_{ed}}{\partial \mathbf{q}_i}$ to the item factor gradient $\frac{\partial \mathcal{L}_{rating}}{\partial \mathbf{q}_i}$.

As [Kabbur, 2015] suggested, gradient updates for model parameters are computed for both rated and non-rated entries of **R**, which is a common practice for the top-N recommendation task. We sample zero entries corresponding to non-rated items on every epoch and add these to the actual training ratings. [Kabbur, 2015] indicates that the ratio between rated and non-rated within the range of 1:3 to 1:5 is sufficient to produce the best model.

When we use the introduced Doc2Vec prediction model together with the nonlinear matrix factorization model MPCFs, we call this joint model **MPCFs-SI**.

2.3.3 MFNN: Neural Network

The second model is an ensemble of a matrix factorization model and a neural network model. This model is called **MFNN**. Inspired by the MPCFs model, it has the ability to incorporate additional user or item data directly as an input to the neural network. As in MPCFs, the matrix factorization is used to model a global preference, while a nonlinear method is used to model local preferences.

The estimated rating $\hat{r}_{ui}$ is given by:

$$\hat{r}_{ui} = \mu + b_i + \mathbf{p}_u \mathbf{q}_i^\intercal + nn(u, i), \tag{2.9}$$

where μ is the global bias, b_i is the item bias corresponding to item i, $\mathbf{p}_u$ is the latent vector associated with user u, $\mathbf{q}_i$ is the latent vector associated with item i and $nn(u, i)$ is a user-item specific local preference component. The user-item specific local preference component $nn(u, i)$ is defined as,

$$nn(u, i) = \sum_{j=1}^{J} (w_j \varphi(\sum_{l=1}^{L} w'_{j,l} x_l)), \tag{2.10}$$

$$\mathbf{x} = (\mathbf{p}_u, \mathbf{q}_i, \mathbf{f}_i), \tag{2.11}$$

$$\varphi(x) = \max(0, x), \tag{2.12}$$

where $w_\cdot$ and $w'_{\cdot,\cdot}$ are weights of the neural network model to be learned, $\mathbf{x}$ is a concatenation of the user latent vector $\mathbf{p}_u$, the item latent vector $\mathbf{q}_i$ and the side information as a Doc2Vec vector $\mathbf{f}_i$, J is the number of neurons in the hidden layer, L is the length of $\mathbf{x}$ and $\varphi(x)$ is the activation function. J, the number of neurons in the hidden layer, is somewhat related to T, the number of user local preferences in the MPCF model. Modeling user-item specific local preferences as a neural network enables the interaction of these local preferences, whereas in MPCF the user local preferences are independent and do not interact. Figure 2.1 shows the neural network in a diagram.

The loss term to minimize for this model is as follows:

$$\mathcal{L} = \frac{1}{2} \sum_{u,i \in R} (r_{ui} - \hat{r}_{ui})^2 + \frac{\lambda_{reg}}{2} (\|\mathbf{P}\|_F^2 + \|\mathbf{Q}\|_F^2 + \|\mathbf{b_i}\|_2^2) + \frac{\lambda_{nnreg}}{2} (\|\mathbf{w}\|_2^2 + \|\mathbf{W}'\|_F^2) \tag{2.13}$$

Like in the previous model MPCFs-SI, we apply AdaGrad to update $\mu, \mathbf{P}, \mathbf{Q}$, and $\mathbf{b_i}$. The neural network was implemented in Theano and we also use AdaGrad to update its parameters $\mathbf{w}$ and $\mathbf{W}'$. The sampling of non-rated items is done the same way as in our first model.

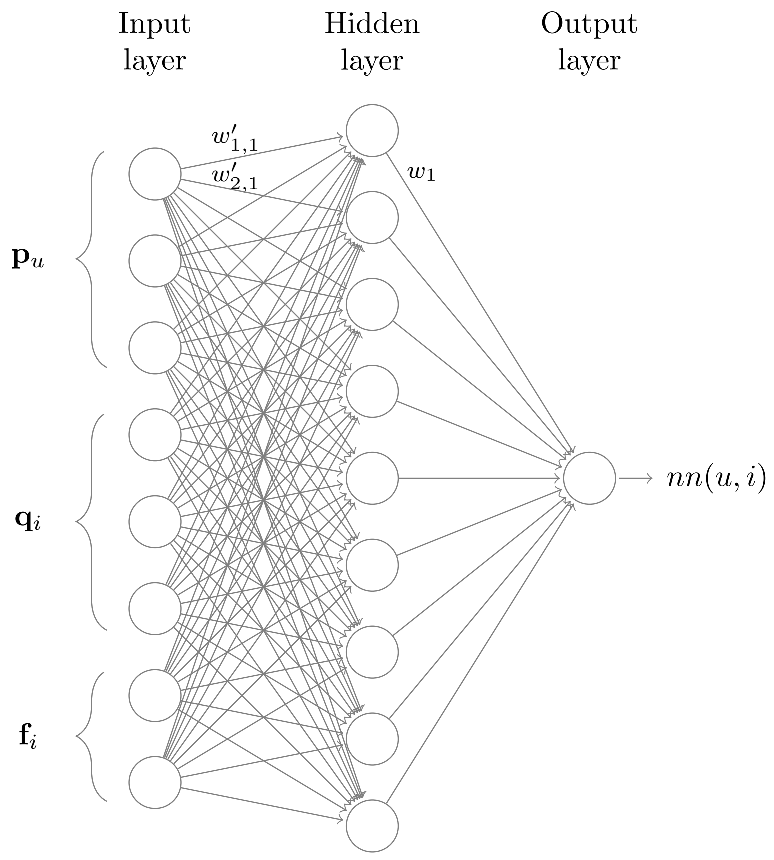

Figure 2.1: MFNN: Neural Network

3

Evaluation

In the previous chapter, we introduced two approaches, MPCFs-SI and MFNN, to incorporate side information into the recommender. The first section of this chapter gives an overview of the used datasets (Section 3.1), followed by the baseline recommender (Section 3.2) and the evaluation methodology (Section 3.3). We then state which performance measures were calculated (Section 3.4). Section 3.5 shows how the proposed models perform compared to state-of-the-art models. We conclude this chapter with an outline of our software setup in Section 3.6.

3.1 Datasets

Our datasets are based on the MovieLens 100k and the MovieLens 1M datasets[1] and call them ML-100k-sub and ML-1M-sub respectively. Only movie ratings were selected based on whether or not their respective movie subtitle was found. Subtitles were downloaded from Opensubtitles[2]. The datasets have rating values from 1 to 5. Converting the ratings into implicit feedback by setting the positive ratings to 1 was applied as it is commonly done for the top-N recommendation task [Kabbur, 2015]. Table 3.1 shows the number of ratings and movies that were selected in the subsets. The rank-size distributions, explaining how many times each movie has been rated and ranked by that value, are presented in Figures 3.1 and 3.2. An analysis of the rank-size distribution has shown that the ML-100k-sub dataset, in particular, has lost a bit of its tail.

Dataset	**# Ratings**	**# Movies**	**# Users**
MovieLens 100k	100'000	1682	943
ML-100k-sub	91'178	1261	943
MovieLens 1M	1'000'000	3883	6040
ML-1M-sub	978'026	3005	6040

Table 3.1: Datasets

[1]MovieLens Datasets - http://grouplens.org/datasets/movielens/
[2]Opensubtitles - http://opensubtitles.org

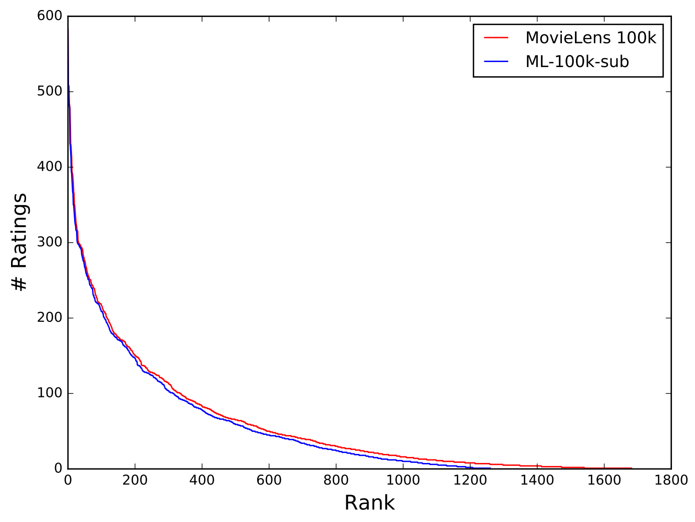

Figure 3.1: Rank-size distribution of the MovieLens 100k and ML-100k-sub datasets

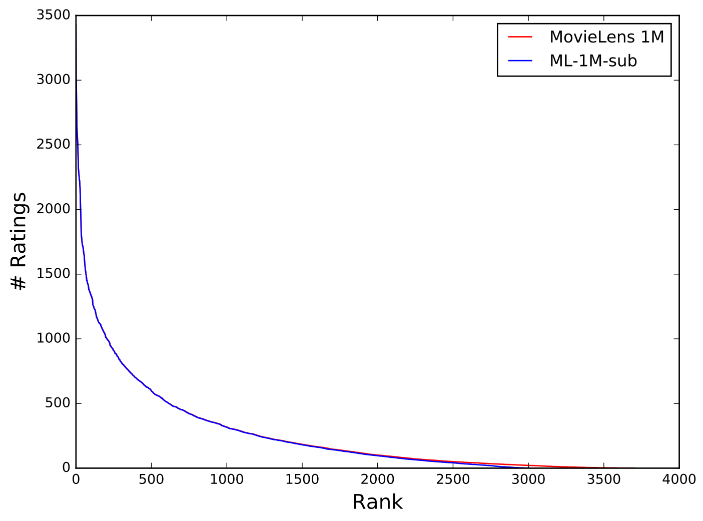

Figure 3.2: Rank-size distribution of the MovieLens 1M and ML-1M-sub datasets

3.2 Baseline

Both the MPCFs-SI (Section 2.3.2) and MFNN (Section 2.3.3) models were compared against three baseline methods. The baseline recommender consists of the nonlinear matrix factorization model MPCFs discussed in Section 2.2, a linear recommender called SLIM [Ning and Karypis, 2011], and another state-of-the-art recommender also based on a latent factor model obtained with matrix factorization, called BPRMF [Rendle et al., 2009].

3.3 Methodology

The standard way of splitting datasets for evaluating recommendation systems is that a percentage of each user's ratings remain in the training set while the rest goes into the test set. However, datasets have been subdivided, such that 70% of each movie's ratings were used for training, and the rest for testing. The rational explanation of this method of splitting the sample is to make the movie data especially sparse. We were also making sure that each user has at least one rating in the test set in order to measure the performance.

For each recommender, a parameter grid was defined and 12 or more parameter sets were randomly drawn from that grid. The parameter set which resulted in the best area under the ROC curve (AUC) for each model was then selected. The best parameter sets are listed in Appendix A.1. The performance of the methods was then evaluated using five different randomly drawn splits in a manner described in the previous paragraph.

Measures were reported after 20 iterations for MPCFs, MPCFs-SI, and BPRMF. The number of iterations for the SLIM model was a hyperparameter. We have found that five epochs give us the best results. Our model MFNN is computationally quite expensive and takes a lot of time. We also ran 20 iterations of the MFNN model on the ML-100k-sub, however, we had to limit the number of iterations of that model for the ML-1M-sub dataset to 10 epochs.

3.4 Performance Measures

We use the area under the ROC curve (AUC) as our main measure to evaluate each approach. Moreover, we have adopted several commonly used measures to evaluate the performance of the recommender. The following measures were calculated: area under the ROC curve (AUC), Precision@20 (P@20), Recall@20 (R@20), Mean Reciprocal Rank (MRR) and Spearman Rank Correlation (SRC). Furthermore, we have carried out an additional measure related to AUC: Instead of recommending items to users and calculate the ranking measure, we were recommending users to items and calculating the rank measure for each item. This is called IAUC.

3.5 Performance Comparison

Measures were defined in Section 3.4 and used to determine the performance of the models introduced in Section 2.3 as well as the baseline recommender. MPCFs-SI performs better on all calculated performance measures on ML-100k-sub compared to the best baseline recommender, which is MPCFs. On ML-1M-sub, MPCFs-SI is superior to MPCFs in terms of AUC, IAUC and Spearman Rank Correlation. Results show that MFNN is inferior to MPCFs on all measures except Precision@20 and Recall@20 on the ML-100k-sub dataset. MFNN is performing at a similar level as MPCFs-SI regarding AUC and has the best Spearman Rank Correlation of all recommender on the bigger ML-1M-sub. The full results are summarized in Table 3.2.

Table 3.3 presents that similar movies have a high cosine similarity (defined in Eq. 2.6) in the document vector space. In fact, the sequels to the listed movies were the most or second most similar movie in that space. Moreover, looking at the item latent vector space, results show that using MPCFs-SI improves the cosine similarity of all similar items compared to using MPCFs. The most dissimilar movies in the document vector space of the first listed movies are also analyzed. MPCFs-SI also separates dissimilars apart and thus decreases the cosine similarity of those movies. MFNN seems to make all listed movies more similar, irregardless of them being similar in the document vector space or not. The best performing models on the ML-1M-sub dataset were used to calculate the cosine similarities in Table 3.3.

One of our goals was also addressing the cold-start problem. Our hypothesis here is that incorporating side information would lead to better recommendations for users who have very few ratings. Figures 3.3 (ML-100k-sub dataset) and 3.6 (ML-1M-sub dataset) show the average AUC for a group of users as users with more training ratings were included into that group.

It is surprising that the average AUC goes down the more users with many training ratings are being included in Figure 3.3. However, this must be caused by an inherent characteristic of the ML-100k-sub dataset as that trend is seen for all recommender. We do not observe that using MPCFs-SI or MFNN improve the AUC for users who have very few ratings on the ML-100k-sub dataset.

On the bigger ML-1M-sub dataset, Figure 3.6 and Figure 3.7 (the zoomed version thereof) indicate that the MFNN model is superior to the other recommender in terms of AUC for users with less than 200 training ratings in particular. However, MPCFs-SI seems to have an advantage for users with less than 10 training ratings.

For the sake of completeness, the progressions of the average of the other defined measures are presented in Figures 3.4 and 3.5 for the ML-100k-sub dataset, and in Figures 3.8 and 3.9 for the ML-1M-sub dataset.

	Method	AUC	IAUC	P@20	R@20	MRR	SRC
ML-100k-sub	Perfect	100.0	100.0	100.0	100.0	1.0	1.0
	MPCFs-SI	**93.76**	**90.83**	28.16	**44.27**	**0.6866**	**0.2005**
	MFNN	93.48	89.74	**28.44**	43.96	0.6741	0.1925
	MPCFs	93.65	90.78	27.79	43.55	0.6770	0.1974
	BPRMF	92.20	68.03	18.00	27.38	0.3703	0.1567
	SLIM	91.45	85.20	23.49	37.75	0.5993	0.1880
ML-1M-sub	Perfect	100.0	100.0	100.0	100.0	1.0	1.0
	MPCFs-SI	92.88	**92.15**	32.11	32.77	0.6941	0.2260
	MFNN	**92.89**	91.73	30.36	29.54	0.6481	**0.2351**
	MPCFs	92.81	92.08	**32.71**	**33.44**	**0.7035**	0.2223
	BPRMF	91.82	67.54	21.23	20.13	0.4430	0.2013
	SLIM	91.23	88.00	28.59	28.76	0.6465	0.2322

Table 3.2: Performance measures. The best result for each measure and dataset is highlighted in bold.

Movie 1	Movie 2	Doc2Vec	MPCFs	MPCFs-SI	MFNN
Free Willy	Free Willy 2	0.8694	0.6915	0.7329	0.8305
Jurassic Park	Jurassic Park 2	0.6796	0.5418	0.6214	0.7440
Scream	Scream 2	0.7514	0.7016	0.7362	0.8325
Species	Species II	0.6831	0.6641	0.7015	0.8203
Star Wars V	Star Wars VI	0.9231	0.6846	0.7650	0.8717
Toy Story	Toy Story 2	0.7529	0.6546	0.6838	0.7929
Free Willy	Mrs. Brown	-0.0141	-0.0366	-0.0927	0.0977
Jurassic Park	Battling Butler	-0.1127	-0.1218	-0.1805	-0.0083
Scream	Kelly's Heroes	-0.0465	-0.1385	-0.1971	-0.0868
Species	Patton	-0.0763	-0.0283	-0.0485	0.0942
Star Wars V	Angela's Ashes	-0.0130	-0.1433	-0.1732	-0.1183
Toy Story	Raise the Red Lantern	-0.0257	-0.0256	-0.0490	0.0593

Table 3.3: Item cosine similarities, measured in the document vector space and the item latent vector space of the according model.

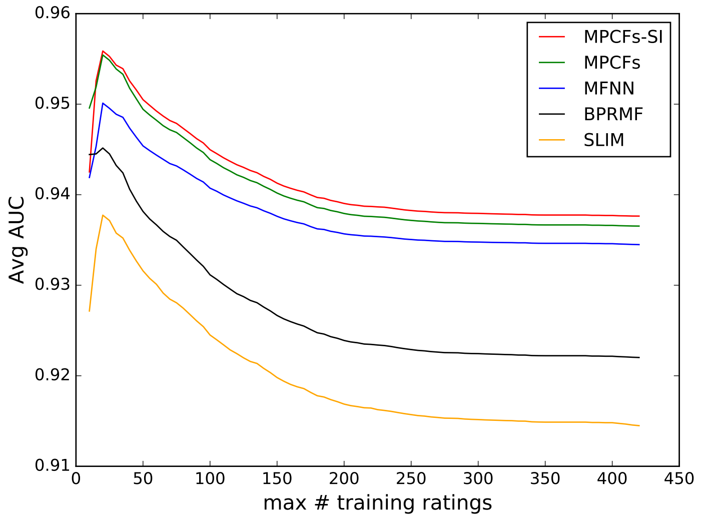

Figure 3.3: Average AUC for a group of users as users with more training ratings were included into that group. The performance is measured on the ML-100k-sub dataset.

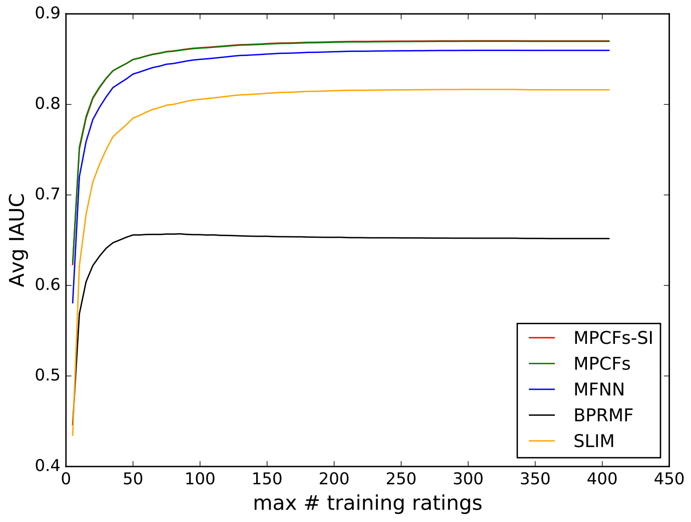

Figure 3.4: Average IAUC for a group of users as users with more training ratings were included into that group. The performance is measured on the ML-100k-sub dataset.

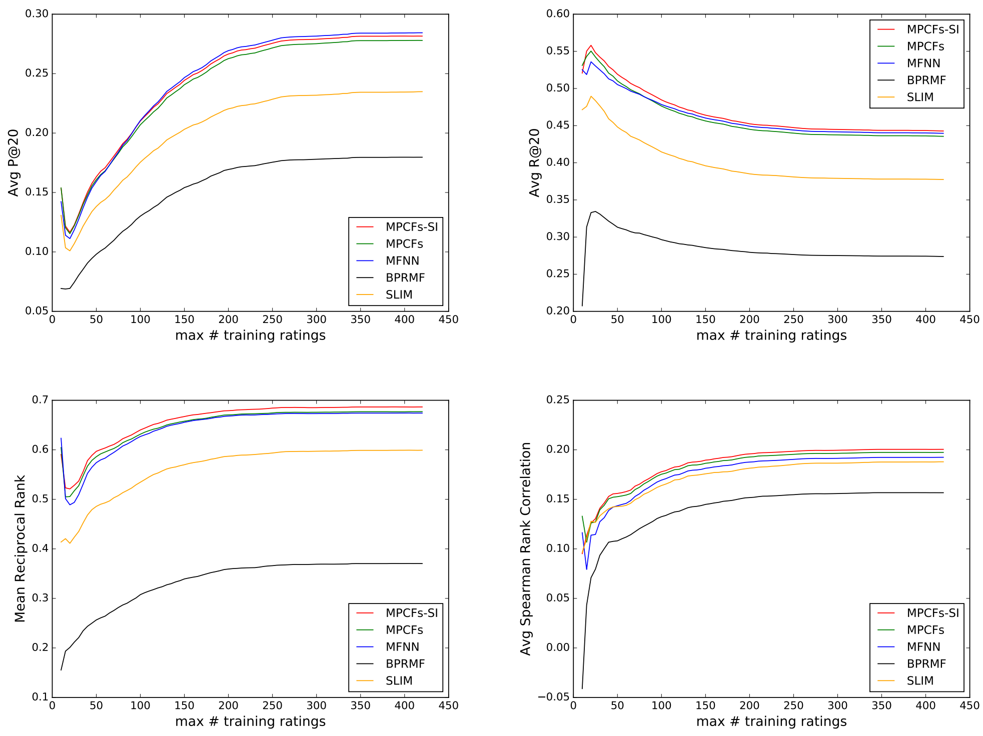

Figure 3.5: Measured on the ML-100k-sub dataset.

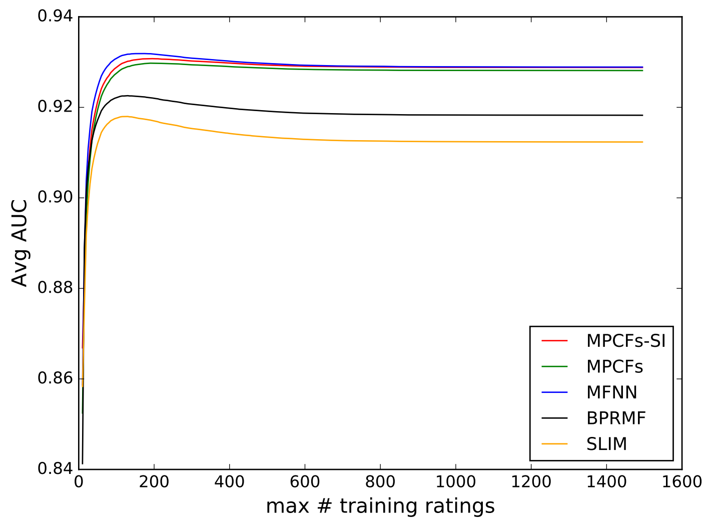

Figure 3.6: Average AUC for a group of users as users with more training ratings were included into that group. The performance is measured on the ML-1M-sub dataset.

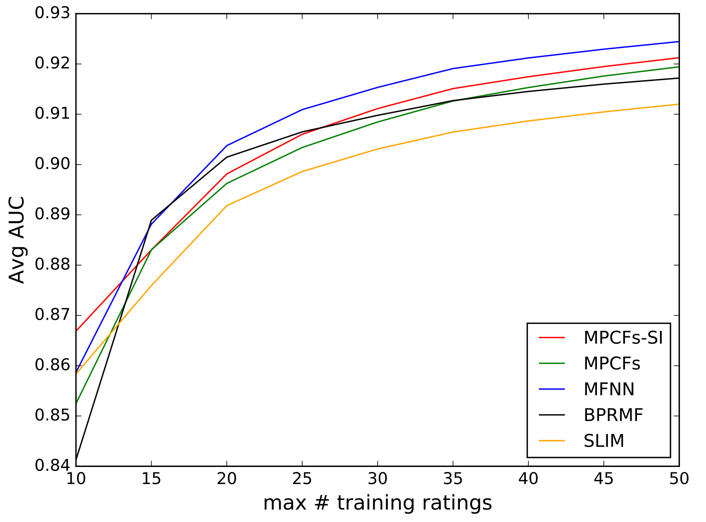

Figure 3.7: Average AUC for a group of users as users with more training ratings were included into that group. The performance is measured on the ML-1M-sub dataset. Zoomed version of Figure 3.6.

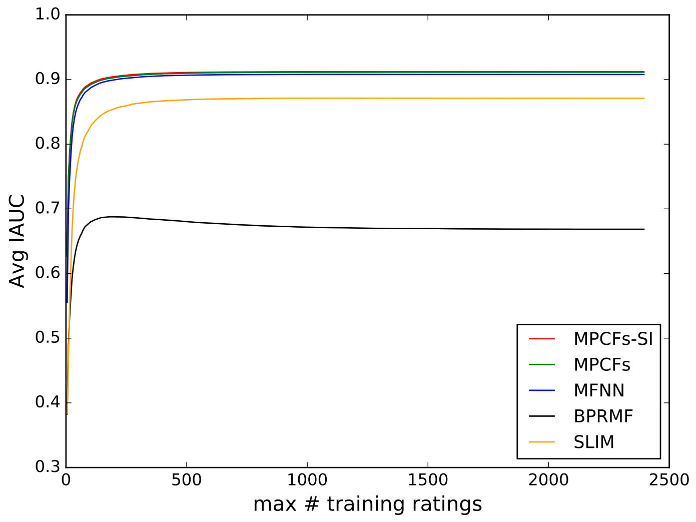

Figure 3.8: Average IAUC for a group of users as users with more training ratings were included into that group. The performance is measured on the ML-1M-sub dataset.

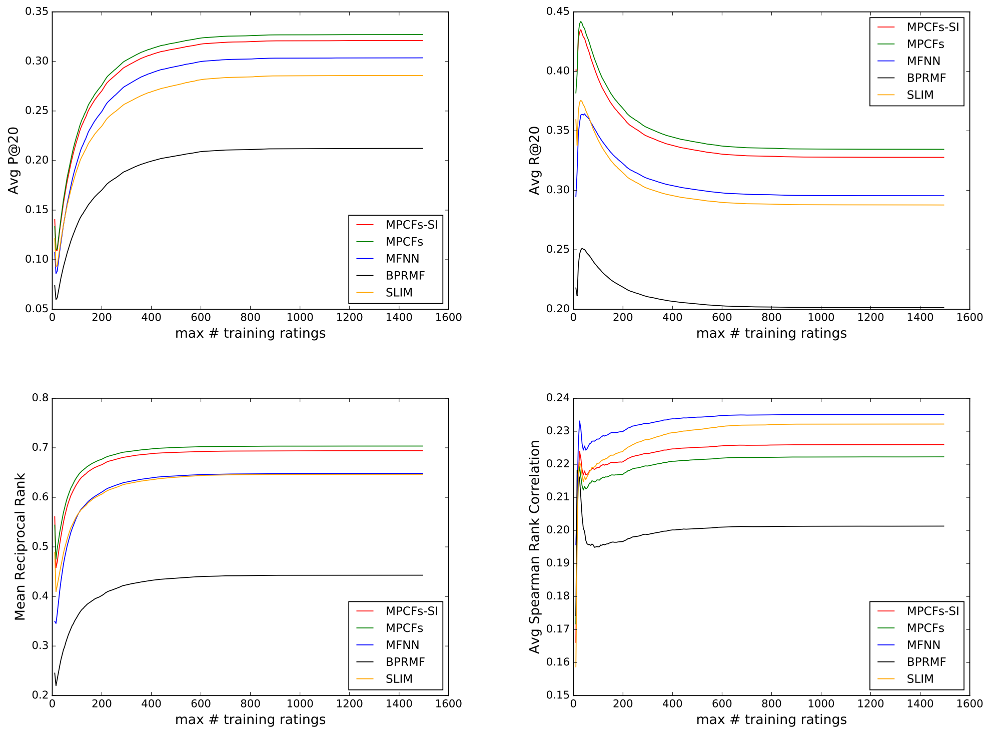

Figure 3.9: Measured on the ML-1M-sub dataset.

3.6 System Setup

A custom software was built in Python. Our two models, MPCFs-SI and MFNN, as well as the baseline recommender MPCFs [Kabbur, 2015] and BPRMF [Rendle et al., 2009] were implemented by us. For the baseline recommender SLIM [Ning and Karypis, 2011], we were able to use parts of an open source implementation by Mendeley[3]. Data handling was done with the Pandas library[4] and we have used Numpy[5] for linear algebra operations. As explained in Section 2.3.1, feature vectors for the movies were extracted from their subtitles with the help of Gensim[6] and NLTK[7]. The affine transformation of the MPCFs-SI model, as well as the neural network of the MFNN model, were implemented with Lasagne[8] and Theano[9]. The source code for this work can be downloaded under https://github.com/marcuniq/bsc-thesis.

[3]Mendeley mrec - https://github.com/Mendeley/mrec
[4]Pandas - http://pandas.pydata.org/
[5]Numpy - http://www.numpy.org/
[6]Gensim - https://radimrehurek.com/gensim/
[7]NLTK - http://www.nltk.org/
[8]Lasagne - https://github.com/Lasagne/Lasagne
[9]Theano - http://deeplearning.net/software/theano/

4

Conclusions

In this work, two models, MPCFs-SI and MFNN, were developed which incorporate side information encoded as feature vectors into recommender systems in order to improve top-N recommendations. MPCFs-SI utilizes the side information to regularize its underlying base recommender, a nonlinear matrix factorization model (MPCFs) developed by [Kabbur, 2015]. MFNN is a combination of a matrix factorization and a neural network. That model uses the additional extra data as an input to the neural network, together with the user and item latent vectors.

Our results have shown that MPCFs-SI is able to outperform the baseline recommender MPCFs on the two MovieLens subsets, ML-100k-sub and ML-1M-sub, for the top-N recommendation task. MFNN is inferior to MPCFs on the ML-100k-sub dataset, however, it was at a similar level of performance as MPCFs-SI on the ML-1M-sub dataset. In particular, MPCFs-SI has an advantage for users with less than 10 training ratings on ML-1M-sub, while MFNN is superior to other recommender for users with less than 200 training ratings also on ML-1M-sub.

Future research efforts should focus on improving the Doc2Vec prediction of the MPCFs-SI model by using a separate weight matrix $\mathbf{G}$ per item. For the MFNN model a similar strategy of having separate weights $\mathbf{w}, \mathbf{W}'$ per user or item could be applied and might lead to further improvements.

References

[Almahairi et al., 2015] Almahairi, A., Kastner, K., Cho, K., and Courville, A. (2015). Learning Distributed Representations from Reviews for Collaborative Filtering. *Proceedings of the 9th ACM Conference on Recommender Systems - RecSys '15*, pages 147–154.

[Bao et al., 2014] Bao, Y., Hui, F., and Zhang, J. (2014). TopicMF: Simultaneously Exploiting Ratings and Reviews for Recommendation. *Aaai*, pages 2–8.

[Carello and Moreno, 2005] Carello, C. and Moreno, M. a. (2005). Why Nonlinear Methods? *Tutorials in contemporary nonlinear methods for the behavioral sciences*, pages 1–25.

[Chapman, 2014] Chapman, A. (2014). Introducing Distributed Word Vectors. url: https://www.kaggle.com/c/word2vec-nlp-tutorial/details/part-2-word-vectors, Last visited 2016-07-08.

[Christoffel, 2014] Christoffel, F. (2014). *Recommending Long-Tail Items with Short Random Walks over the User-Item-Feedback Graph.* PhD thesis.

[Goldberg and Levy, 2014] Goldberg, Y. and Levy, O. (2014). word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. (2):1–5.

[Ilin and Raiko, 2010] Ilin, A. and Raiko, T. (2010). Practical approaches to principal component analysis in the presence of missing values. *Journal of Machine Learning Research*, 11:1957–2000.

[Kabbur, 2015] Kabbur, S. (2015). *Machine Learning Methods for Recommender Systems.* PhD thesis, University of Minnesota.

[Kabbur and Karypis, 2014] Kabbur, S. and Karypis, G. (2014). NLMF: NonLinear Matrix Factorization Methods for Top-N Recommender Systems. *2014 IEEE International Conference on Data Mining Workshop*, (ii):167–174.

[Le and Mikolov, 2014] Le, Q. and Mikolov, T. (2014). Distributed Representations of Sentences and Documents. *International Conference on Machine Learning - ICML 2014*, 32:1188–1196.

[McAuley and Leskovec, 2013] McAuley, J. and Leskovec, J. (2013). Hidden factors and hidden topics: understanding rating dimensions with review text. *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, pages 165–172.

[Ning and Karypis, 2011] Ning, X. and Karypis, G. (2011). SLIM: Sparse LInear Methods for top-N recommender systems. *Proceedings - IEEE International Conference on Data Mining, ICDM*, pages 497–506.

[Qiu, 2015] Qiu, L. (2015). Sentiment Analysis using Doc2Vec. url: https://github.com/linanqiu/word2vec-sentiments/blob/master/word2vec-sentiment.ipynb, Last visited 2016-07-08.

[Rendle et al., 2009] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-thieme, L. (2009). BPR : Bayesian Personalized Ranking from Implicit Feedback. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, cs.LG:452–461.

[Ricci et al., 2011] Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*, chapter Introducti, page 842. Springer US.

[Rong, 2014] Rong, X. (2014). word2vec Parameter Learning Explained. pages 1–20.

[Weston et al., 2013] Weston, J., Weiss, R. J., Yee, H., and Bruno, S. (2013). Nonlinear Latent Factorization by Embedding Multiple User Interests. *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 65–68.

A

Appendix

A.1 Parameters

A.1.1 Doc2Vec

distributed memory: 0 (= distributed bag-of-words: 1), size: 50, hierarchical sampling: 0, negative sampling factor: 5, window: 8, minimum word count: 2, number of epochs: 20, alpha: 0.025, minimum alpha: 0.025, alpha decay: 0.0005

A.1.2 MPCFs-SI

ML-100k-sub

Matrix factorization model:
learning rate: 0.01, learning rate decay: 0.02, number of epochs: 20, number of latent factors k: 128, number of user local preferences T: 4, regularization parameter λ_{reg}: 0.01, zero sample factor: 5
Doc2Vec prediction model:
learning rate: 0.03, learning rate decay: 0.02, regularization parameter λ_{reg}: 0.01, balancing parameter λ_{ed}: 0.001, balancing parameter λ_{cos}: 0.7

ML-1M-sub

Matrix factorization model:
learning rate: 0.03, learning rate decay: 0.02, number of epochs: 20, number of latent factors k: 96, number of user local preferences T: 2, regularization parameter λ_{reg}: 0.01, zero sample factor: 5
Doc2Vec prediction model:
learning rate: 0.03, learning rate decay: 0.02, regularization parameter λ_{reg}: 0.01, balancing parameter λ_{ed}: 0.0003, balancing parameter λ_{cos}: 2

A.1.3 MFNN

ML-100k-sub

Matrix factorization model:
learning rate: 0.03, learning rate decay: 0.02, number of epochs: 20, number of latent factors k: 96, regularization parameter λ_{reg}: 0.01, zero sample factor: 5
Neural network model:
layer dimensions: [242, 16, 8, 1], learning rate: 0.03, learning rate decay: 0.02, regularization parameter λ_{nnreg}: 0.003

ML-1M-sub

Matrix factorization model:
learning rate: 0.06, learning rate decay: 0.02, number of epochs: 10, number of latent factors k: 96, regularization parameter λ_{reg}: 0.01, zero sample factor: 5
Neural network model:
layer dimensions: [242, 4, 1], learning rate: 0.06, learning rate decay: 0.02, regularization parameter λ_{nnreg}: 0.01

A.1.4 MPCFs

ML-100k-sub

learning rate: 0.01, learning rate decay: 0.02, number of epochs: 20, number of latent factors k: 128, number of user local preferences T: 4, regularization parameter λ_{reg}: 0.01, zero sample factor: 5

ML-1M-sub

learning rate: 0.03, learning rate decay: 0.02, number of epochs: 20, number of latent factors k: 128, number of user local preferences T: 2, regularization parameter λ_{reg}: 0.01, zero sample factor: 5

A.1.5 BPRMF

ML-100k-sub

learning rate: 0.03, learning rate decay: 0.02, number of epochs: 20, number of latent factors: 128, regularization parameter: 0.01, triplet sample factor: 5

ML-1M-sub

learning rate: 0.03, learning rate decay: 0.02, number of epochs: 20, number of latent factors: 128, regularization parameter: 0.001, triplet sample factor: 5

A.1.6 SLIM

ML-100k-sub

number of epochs: 5, fit intercept: false, ignore negative weights: true, l_1 regularization: 0.003, l_2 regularization: 0.00003

ML-1M-sub

number of epochs: 5, fit intercept: true, ignore negative weights: true, l_1 regularization: 0.001, l_2 regularization: 0.003

List of Figures

List of Tables