Improvement of Recommendation Accuracy by Integrating User Demographic Information

Shuqi Wu

A Dissertation

Presented to the University of Dublin, Trinity College

in partial fulfilment of the requirements for the degree of

Master of Science in Computer Science (Data Science)

Supervisor: Dr. Bahman Honari

08 2019

Declaration

Permission to Lend and/or Copy

Acknowledgments

First and foremost, I would like to thank my parents, Jiayong Wu and Lilan Wu, who love me, believe me and support me to complete my Master degree at Trinity College Dublin.

I would like to express my sincere gratitude to my supervisor, Prof. Bahman Honari for all the support and help he has given to me through out my thesis. Then would like to thank my second reader who taken out some of his time and gave me suggestions for improving my thesis during my presentation.

All of the staff and faculty members in the School of Computer Science and Statistics, for imparting their vast knowledge to me over this year.

Last but not the least, I would like to thank all my friends, and classmates, for making my days the most memorable of my life.

Shuqi Wu

University of Dublin, Trinity College
08 2019

Improvement of Recommendation Accuracy by Integrating User Demographic Information

Abstract

Shuqi Wu, Master of Science in Computer Science
University of Dublin, Trinity College, 2019

Supervisor: Dr. Bahman Honari

Recommender systems has taken more and more places in our lives, recommending goods or services that satisfy the original expectations of users.Since the first research paper came up in the mid 1990s, attempts of implementing different algorithms to provide more personalized items in order to improve the prediction accuracy can be found in the literature. However, as far as I known, most papers about recommender systems so far are only based on users' previous preferences - ratings, neglecting the user demographic information, such as age, gender,occupation. Theoretically, recommendations made with respect to different users characteristics would be more accurate and personalized. That leads to my research question:

to build a recommender system that is using the user demographic information and test if this recommender systems performance would be improved by using the user demographic information.

The baseline algorithm is based on Collaborative Filtering algorithm combining with matrix factorization technique, which was used to solve the data sparsity problem.

Movielens one million dataset was chosen as the experiment dataset as it contains the complete user demographic information dataset (predictors:x) and the traget value (predicted value:Y), ratings, that can used for the comparison between different experiments. A recommender system's performance is measured by mean square error(mse), the comparison between the predicted ratings and the true ratings. The overall experiment was processed as follows:

- Experiment 1: Train the ratings dataset without user demographic information and predict users ratings towards different movies. Use the predicted ratings for the test set to get the overall mse as the evaluation result.
- Experiment 2: Train the datset with user demographic information and predict user ratings towards different movies. Use the predicted ratings for the test set to get overall mse as the evaluation result. However, with the complexity of user demographic information, Convolution Neurall Network was added to train the textual data.
- Experiment 3: Train the datset with two dataset in Experiment 1 and Experiment 2 using User-based Collaborative Filtering algorithm, one added experiment to prove that the improvement in Experiment 2 was resulted from the user demographic dataset. The precision was used as the evaluation matrix.

The mse results of experiment 1 and experiment 2 show that the recommender systems performance by using user demographic information(0.661) implemented with Convolution Neural Network has improved around 15% compared to the experiment without user demographic information(0.770). The Experiment 3 was designed to prove that the improvement of experiment 2 was made by user demographic instead of the Convolution Neural Network algorithm. The precision of the test with user demographic information(46.4%) is 17% higher than that without user demographic information(38.5%). Thus, the above experiments show that the use of user demographic information can improve the overall recommender systems performance.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

During the last few years, recommender systems have emerged to deal with enormous amount of data in online platforms and have taken more and more places in our lives. It has been widely used in different areas: for web-based recommender systems, suggesting products or items to users that could interest them; for online advertisement systems, recommending users the right contents that might match their preferences. These systems are generating recommendations for users based on users' previous behaviors. By providing more personalized recommendations to improve the recommender systems performance, the user experience and loyalty can be improved.

First research paper about recommender systems was published in 1990 [2] by Jussi Karlgren. Since then, the researches about recommender systems have never stopped. Research groups are organized these years to study for different algorithms. Commercial companies are running competitions to award for a higher-performance recommender system model. Challenges and problems are arisen and discussed throughout the years. Literature surveys on available research papers are published year by year.

1.2 Motivation

As the increasing performance of recommender systems, the study of it is not restricted to the algorithm itself. The training dataset is one of the other aspects we could focus on. Basically, a recommender system is an extensive class of web applications that aims to predict the users responses to a new product based on users' previous preferences, that is ratings. Most research papers about recommender systems are based only on ratings dataset, neglecting the other user demographic information.

Take an example of MovieLens dataset. MovieLens dataset is of the widely used dataset for the recommender systems research purposes. It is collected from MovieLens online movie recommender system. It contains approximately one million ratings from about 6,000 users on around 4,000 movies. MovieLens dataset are structured as three separated files, ratings dataset, users demographic inforamtion dataset, and the movies dataset. The ratings dataset can be seen as the users previous preferences when making recommendations. The users dataset saved all the user demographic information, such as age, gender, and occupation. The last movies dataset records the movie titles and genres.

According to the analysis about MovieLens dataset, I found that different gender groups have different preferences to movies. Female users' top 5 preferences to movies are [Drama(22%), Comedy(14%), Romance(12%), Action(10%), Thriller(9%)] while male users' references are [Drama(20%), Comedy(12%), Action(12%), Thriller(10%), Romance(9%)]. The same results came up with the different age groups and the different occupations. This analysis shows that the user demographic information are having some relationships with one user's preferences. This result based on the MovieLens dataset have motivated my research question.

1.3 Research Question

The research question of my dissertation is:

to build a recommender system that is using the user demographic information and test if this recommender systems performance would be improved by using the user demographic information.

1.4 Research Objective

The research objective put forth to address the research questions are:

- Pre-processing the MovieLens datasets and converting all the data types to be suitable for training for different experiments.
- Experiment 1: Constructing an algorithm to experiment the recommender systems performance without user demographic information by using Collaborative Filtering algorithm.
- Experiment 2: Constructing a novel algorithm to experiment the recommender systems performance with user demographic information by using Collaborative Filtering algorithm combined with Convolution Neural Network.
- Experiment 3: Train the dataset with two dataset in Experiment 1 and Experiment 2 using User-based Collaborative Filtering algorithm to prove that the results of experiment 2 is resulted from the user demographic dataset. Use the predicted ratings for the test set to get the precision as the evaluation result.
- Executing three models to calculate the mean square error for the overall recommender system.
- Analyze and compare above results and figure out if the recommender systems performance would be affected by the user demographic dataset.

1.5 Thesis Challenges

- The pre-processing of training dataset - to convert all the data structures to be able for the training with Convolution Neural Network.
- The implementation of all three models and evaluation measurement can only be designed by myself and compared within in this thesis, other algorithms of other papers cannot be used and compared within this thesis. Because different people have different ways of implementing even one same algorithm, thus little difference within one algorithm may result in different result.
- Building a new recommender system by using Convolution Neural Network and testing the parameters for the model.

- Generating the movie genres features because each movie may belong to different genres and each genre contains different features.
- Analysis of the different movie preferences with respect to different user demographic information.

1.6 Thesis Overview

Convolution Neural Network has been widely used for handwritten recognition and image classification since the LeNet-5 model proposed by Yann LeCun et.al [3]. Applying Convolution Neural Network to MovieLens one million dataset integrating the user demographic information is a novel approach in the area of recommender systems. Collaborative Filtering algorithm is one of the most common used algorithms and matrix factorization technique was also chosen for the implementation of this thesis as its effectiveness of solving the data sparsity problem along with Collaborative Filtering algorithm[4]. Thus, the combination of Collaborative Filtering algorithm and matrix factorization technique was chosen to be the baseline algorithm for Experiment 1. Second experiment was operated by using baseline algorithm and parts of the calculation was made by Convolution Neural Network due to the complexity of textual user demographic datase. Experiment 3 was designed and aims to prove that the performance improvement made by Experiment 2 was caused by the dataset instead of the algorithm. Thus, Experiment 3 was operated by using Collaborative Filtering and the same user demographic dataset in Experiment 2.

The MovieLens one million dataset used in this thesis was collected from MovieLens movie recommender system over various periods of time by GroupLens Research, acknowledging the permissions of usage for the research purposes. This dataset contains approximately one million ratings from about 6,000 users on around 4,000 movies. All three experiments are predicting user ratings for different items and the evaluation is measured by the mean square error(mse) algorithm to compare the users' predicted ratings and the true ratings. The mse results for Experiment 2(0.661) with user demographic information is around 15% lower than that of Experiment 1 without user demographic information(0.770). The precision of the test with user demographic information(46.3%) is 20% higher than that without user demographic information(38.5%).

That confirms that the recommender systems implemented with user demographic information are providing higher performance than that of without user demographic information (with ratings dataset).

1.7 Thesis Structure

The thesis is organized as follows: Chapter 2 introduced the related work; Chapter 3 explains the methodologies and implementations for three experiments; the results will be discussed in Chapter 4; Chapter 5 will give the overall conclusion and the future works.

1.8 Keywords

Recommender Systems, Collaborative Filtering, Matrix Factorization, Gradient Descent, Convolution Neural Network, Mean Square Error, Similarity Matrix

Chapter 2

Literature Review

In this chapter, the works related to the methodologies used in this dissertation will be introduced and explained.

2.1 Recommender Systems

With the growing of various selections in our lives, customers are making choices at almost every seconds. In another words, customers are drowning in a explosive information due to the high availability of products or services. As results, this availability is causing the troublesome and time-consuming issue. Therefore, filtering useful information and helping the customers to make the best choice that could meet the expectations with them becomes more and more important. Recommender systems, designed as a filtering tool, are now implemented by different business companies to filter useful information and speed up the whole searching process for users interested products. Thus, the increasing performance of recommender system, in return, are increasing the sales and user experience by predicting more personalized items and meeting the higher satisfaction of customers. The importance of using relevant efficient algorithms to provide a higher accuracy within recommender systems is non-negligible.

Jussi Karlgren published the first research paper in recommender systems in 1990 [2] at Columbia University in a specialized report. Since then, researches in this area became diversified and various approaches were introduced to present higher recommendation accuracy. The basic idea of recommender systems is acting as a information filtering tools to filter all useful information in a effective way. The recommendations are usually made based on users previous preferences. Different algorithms are using different techniques for the implementation. According to those differences, recommender systems can be classified into different categories.

2.2 Collaborative Filtering

One recent survey discussing of recommender system types and classification produced by Akhil P.v and Dr. Shelbi Joseph [2] classifies the recommender system algorithms into three categories, Collaborative filtering, content based filtering and hybrid filtering. Figure 4.5 shows the classifation of recommender systems.

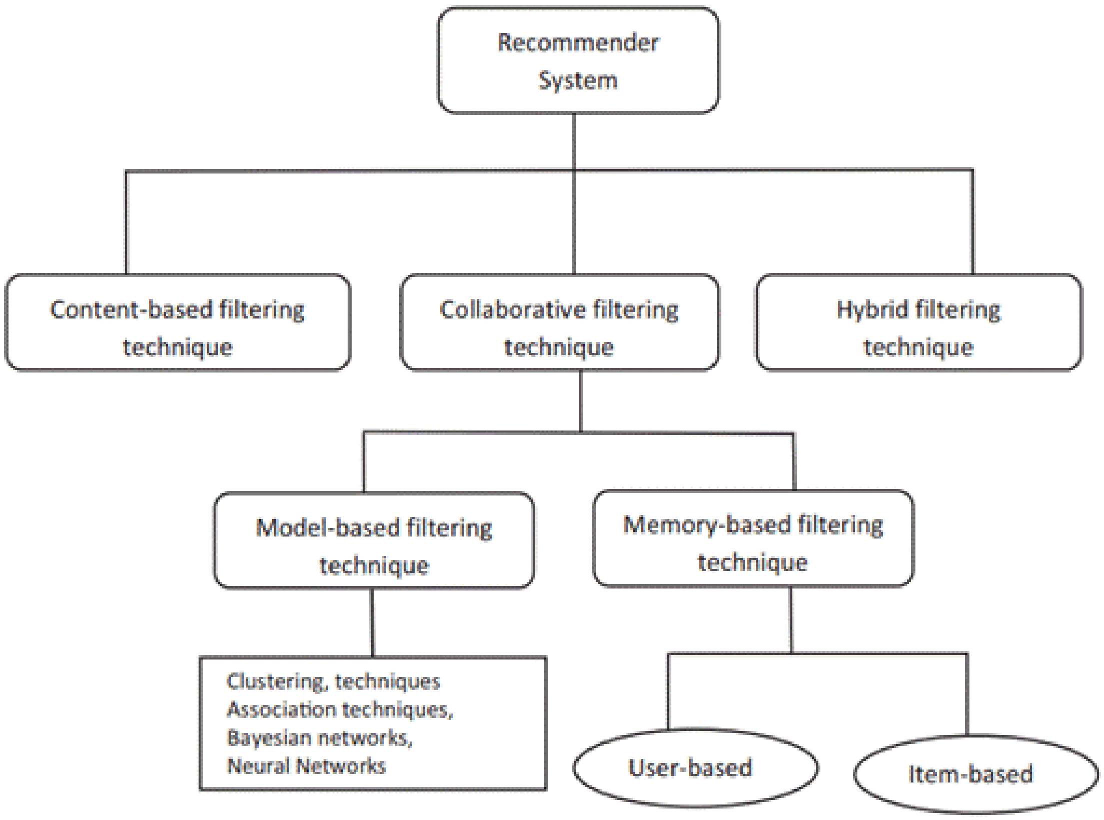

Recommender System
Content-based filtering technique
Collaborative filtering technique
Hybrid filtering technique
Model-based filtering technique
Memory-based filtering technique
Clustering, techniques
Association techniques,
Bayesian networks,
Neural Networks
User-based
Item-based

Figure 2.1: Classification of Recommender System approaches.[2]

Collaborative filtering is a method of predicting the user responses to a new recommended good or service based on users previous preferences. In most cases, users preferences are represented by the ratings to products. User-based Collaborative filtering technique filters the information by using other similar users preferences. Basically, the similar users are selected and filtered by their rating patterns [5]. For instance, if user A and user B both rated item 3 for the rating 2, we can sat that they are sharing the same or similar rating patterns. Therefore, user A and user B are two similar users. Based on user A's rated items, we can recommend those items to user B. This is one common approach called User-Based Collaborative filtering. The predicted rating of the item user A gives to item i can be calculated as an aggregation of other similar users' ratings of items. It can be mathematically represented as follows:

$$r_{u,i} = agg_{u' \in U} r_{u',i} \tag{2.1}$$

where U denotes the set of similar users who are sharing the similar rating patterns and at the same time who have rated the item i. The aggregation formular is:

$$r_{u,i} = N \sum_{u' \in U} sim(u, u') r_{u',i} \tag{2.2}$$

where N is a factor to normalize the whole model, it is defined as

$$1 / \sum_{u' \in U} |sim(u, u')|$$

As for User-Based Collaborative filtering, recommendations are generated to a user based on the evaluation of rated items by his similar users. Thus the similarity matrices between users should be calculated. The similarities between users can be measured by taking weighted average of all the ratings produced by other similar users. Different similarity algorithms[6], such as k-nearest neighborhoods, Jaccard correlation, Pearson correlation, vector cosine etc. Two common techniques are the Pearson correlation similarity and cosine similarity:

- The Pearson correlation similarity of user A and user B can be defied as:

$$sim(A, B) = \frac{\sum_{i \in I_{AB}} (r_{A,i} - \bar{r_A})(r_{B,i} - \bar{r_B})}{\sqrt{\sum_{i \in I_{AB}} (r_{A,i} - \bar{r_A})^2} \sqrt{\sum_{i \in I_{AB}} (r_{B,i} - \bar{r_B})^2}} \tag{2.3}$$

where

$$I_{AB}$$

is the set of items rated by bother user A and user B.

- The cosine similarity of user A and user B can be defied as:

$$sim(A,B) = cos(\vec{A},\vec{B}) = \frac{\vec{A}\cdot\vec{B}}{||\vec{A}|| \times ||\vec{B}||} = \frac{\sum_{i\in I_{AB}} r_{A,i} r_{B,i}}{\sqrt{\sum_{i\in I_A}(r_{A,i})^2}\sqrt{\sum_{i\in I_B}(r_{B,i})^2}} \tag{2.4}$$

Cosine approach was used in my experiment. Overall, user-based collaborative filtering is operated of three steps:

- Look for users who share the similar rating patterns with the user whom the prediction is for by using the above consine similarity algorithms. Similarity users can be one, two, or even thousands, but all with different similarity factors according to their different rating patterns.
- Use the ratings rated by similar users found in step 1 and calculate predicted ratings the active user (users whom those prediction are for) would give to items.
- Recommend top-k items from those sorted predicted items to active user.

2.3 Matrix Factorization

Challenges within the collaborative filtering algorithm are discussed as well. One paper published by Miha et al. [7] in 2006 clearly stated the data sparsity problem in the Collaborative Filtering. The data sparsity problem indicates that the amount of missing ratings for items is large due to the large commercial dataset itself. As the result, the user-item matrix would be extremely large and sparse. Because the number of rated items for most users is extremely small compared to the total item numbers. Therefore, the accuracy of locating the similar users would be very low due to the lack of user preferences information. One typical problem following by the data sparsity is the cold start problem. Cold start problem refers to a situation that the system fail to capture users preferences when a new user is added to the system thus caused a low accurate prediction.

Matrix factorization as an extension of Collaborative Filtering algorithm is used to solve the data sparsity problem. In 2006, Simon Funk reported the initial effective work of this technique in his blog after the Netflix Prize and in 2009 Yehuda Koren et al.[8] published a paper and detailedly described the matrix factorization technique. The basic idea of it is to decompose or factorize the user-item matrix to two smaller dimensional matrices, user matrix and item matrix. Each matrix contains the related hidden latent factors or features. For example, in a movie recommender system context, each movie is represented by ten values describing how much that movie exemplifies each aspect, and similarly each user is represented by ten values referring how much they like each aspect. The dot product of these two matrices can show how much each user prefer each item. Simply, we can define those values as the weights of genres. Each movie has different weights towards different genres and each user has different preferences to different genres. E.g., for movie 'The lion king', it contains:

$$(action = 0.5, romance = 0.8, drama = 0.6, ...)$$

For user A, his preferences to movies is represented as:

$$(action = 2, romance = 8, drama = 4, ...)$$

Therefor, we can get how user A like the movie 'The lion king' by combining those two information together:

$$2 \times 0.5 + 8 \times 0.8 + 4 \times 0.6 + ...$$

Matrix Factorization can be mathematically represented as follows:

Assume that each user u is combined with a user latent factor $P_i \in R^d$ and each item i is combined with a item latent factor $Q_j \in R^d$. The latent factor vector P_i and Q_j describes the preferences of user i and item j. Based on this, the latent factor model uses observed ratings to learn the latent feature vector to predict each rating as an implicit outcome of the latent factor vector[4]. The user-item matrix is the dot product of the above two matrices.

$$R_{i,j} = P^T Q \tag{2.5}$$

2.4 Evaluation Metrics

Evaluation matrices are different according to different learning purposes. Same evaluation matrix might perform differently via supervised learning or unsupervised learning. As for the recommender systems, it can be seen as a classification problem. Thus, the results can be measured by Presion, recall, accuracy or the mean square of errors(mse) to know the accuracy of prediction of the model. In this dissertation, I am using mse as the evaluation measurement.

2.4.1 Precision, Recall, and Accuracy

Precision, recall and accuracy are three matrices used of measuring the percentage of the correct predictions made by the recommeder systems model[9]. In the recommender system context, those evaluation matrices only focus on if one user have rated one item or not. The actual rating figure is not calculated. Some basic terms are:

- Positive: The ground truth is positive (e.g. User 1 rated for item 2).
- Negative: The ground truth is negative (e.g. User 1 did not rate for item 2).
- True Positive: The prediction is positive; the ground truth is positive (e.g. The prediction is user 1 rated for item 2; The ground truth is user 1 rated for item 2).
- False Positive: The prediction is negative; the ground truth is positive (e.g. The prediction is user 1 did not rate for item 2; The ground truth is user 1 rated for item 2).
- True Negative: The prediction is positive; the ground truth is negative (e.g. The prediction is user 1 rated for item 2; The ground truth is user 1 did not rate for item 2).
- False Negative: The prediction is negative; the ground truth is negative (e.g. The prediction is user 1 did not rate for item 2; The ground truth is user 1 did not rate for item 2).

Precision is used to measure how many positive ground truth were actual predicted correctly.

$$Precision = \frac{TruePositive}{True\ Positive + False\ Positive} = \frac{correct\ predictions}{all\ predictions} \tag{2.6}$$

Recall is used to retrieve the percentage of related information that are successfully retrieved. Recall can be calculated by all true positive predictions and all positive results, referring to the numbers of total relevant results that are correctly categorized by the model.

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} = \frac{predicted\ to\ bepositive}{all\ positive\ observations} \tag{2.7}$$

Accuracy is the number of true positive and true negative divided by total numbers, measuring the proportion of all predictions that are correct. It is used to measure how accuracy the model is.

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \tag{2.8}$$

Thus,

$$Accuracy = \frac{correct\ predictions}{all\ predictions} \tag{2.9}$$

The precision and recall both can express the accuracy of the model.

2.4.2 Mean Square Error

Last common used evaluation matrix is called the mean square error. It is a measurement of the average of the squares of the errors between the estimated values and what is estimated[10]. It calculates the errors of exact rating figures rated by the exact user for the exact item. In this dissertation, it is used to measure the difference between the predicted ratings by the system and the users true rating. It can also used within the process of training, user feature matrix P and the item feature matrix Q for matrix factorization can be retrieved by optimizing the mean square error.

Mathematically mean square error can be described as:

$$L = argmin_{p^*,q^*} \sum_{(u,i)\in K} (r_{ui} - (q_i)^T p_u)^2 + \lambda(||q_i||^2 + ||p^u||^2) \tag{2.10}$$

We add regularization term on the right side of the equation in case we find to many latent factors and overfit our model.

In order to solve the above optimization, gradient descent technique was chosen. Gradient descent is one of the most popular algorithms to perform optimization. It is an iterative process that finds the local minimum of a cost function. Take the partial derivative respect to p and q to optimize those values. The final matrix p and q iterated by the given steps are our final results.

$$q_i = q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i) \tag{2.11}$$

$$p_u = p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u) \tag{2.12}$$

2.5 Convolution Neural Network

Convolution Neural Network as a class of deep neural network is widely known in the area of visual imagery [11] and speech precognition [12]. Convolution Neural Network takes an inpyt image, process it and classify it to a certain categories.Typically, the input image will be processed through serious of convolution layers with filters, pooling layers, fully connected layers and finally end up with Softmax function to classify the input image with probabilistic values 0 or 1. One classic model is called LeNet-5, an architecture propsed by Lecun, Y. et al. [3] used for handwritten and machine-printed character recognition in 1998 to extract feathers within hand writing as the hand writing contains too less information.

This model consists of seven layers, two convolution layers, two average pooling layers, two full connection layers and one Softmax classifier. More details will be described as follows.

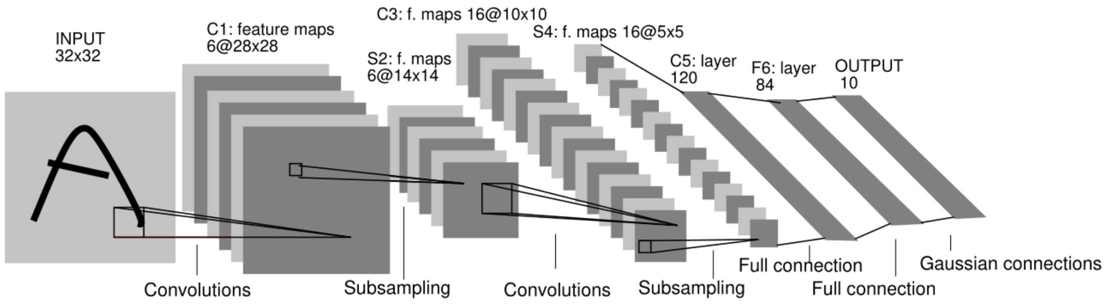


Figure 2.2: Architecture of LeNet-5 [3]

2.5.1 Convolution layer

Convolution layer is the first layer to extract features from the input image by a set of filters. The size of the filters are smaller than the input matrix. Each filter will be automatically assign some random numbers.Those filters are slide across the width and height of the input matrix. The dot products between the filters and the input matrix are computed at every spatial position. Take an example of an 5×5 Input Matrix with 3×3 filter,the output is calculated to be a 3×3 matrix. The [3,3] item of output matrix can be mathematically calculated as follows:

$$1 \times 1 + 1 \times 0 + 1 \times 1 + 1 \times 0 + 1 \times 1 + 0 \times 0 + 1 \times 1 + 0 \times 0 + 0 \times 1 = 4 \qquad (2.13)$$

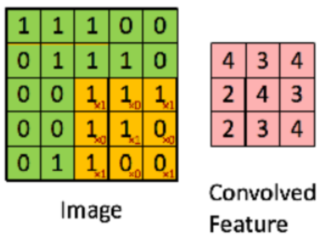


Figure 2.3: 5×5 Input Matrix with 3×3 filter

In LeNet-5,the input is a 32×32 written image with values 0 and 1. After filtered by 6 5×5 filters, the output is 6 28×28 matrix. Then the convolution of 32×32

image matrix multiplies with 5×5 filter matrix which is called 'feature map'.

Convolution with different filters can capture different features[3].

In some cases, Rectified Linear Unit (ReLU) is applied to the input matrix, which aims to transfer all negative values to be non-negative. Other non-linear function such as sigmoid, which is widely used in the Supported Vector Machine (SVM), can also be used for the operation.

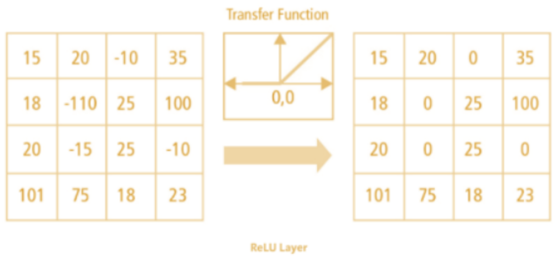


Figure 2.4: ReLU operation

2.5.2 Pooling layer

Sensitivity to the location of features is one of the problems caused after convolution layer. Pooling layers effectively reduce the size of the output feature maps when the input matrix is too large. It reduces the dimension of each map and enables the output matrix to capture the most important features. Therefore, spatial pooling is also called subsampling or downsampling. There are several different types of pooling so far:

- Max-pooling
- Average-pooling
- Sum-pooling

Max-pooling is the most common approach. In LeNet-5 model, after the 2×2 max-pooling layer, six output feature map becomes six 14×14 matrices.

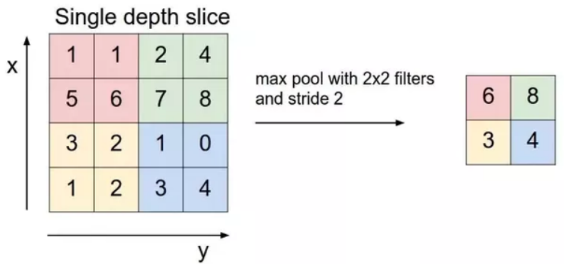


Figure 2.5: Max-pooling

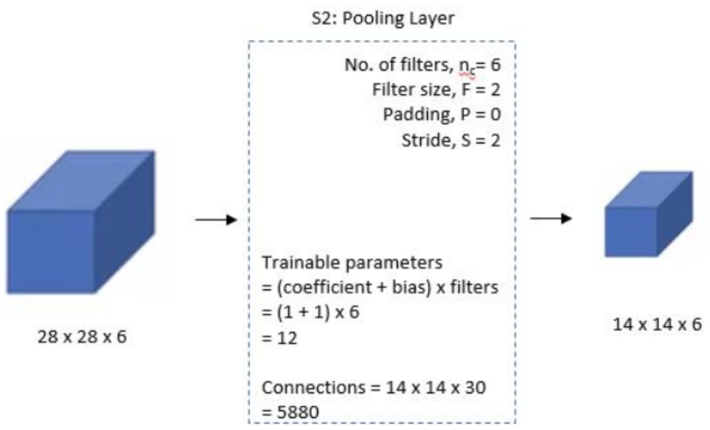


Figure 2.6: Pooling Layer

2.5.3 Fully Connected Layer

The layer that flat all feature maps into a vector is called fully connected layer. In LeNet-5 model, the fifth layer is a fully connected layer with 120 1×1 feature maps. The sixth layer is also a fully connected layer with the 84-feature output feature map.

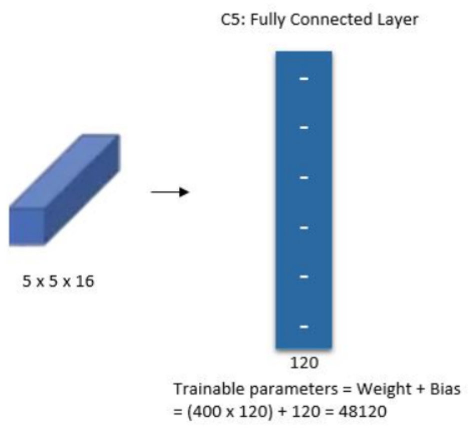

C5: Fully Connected Layer
5 x 5 x 16
120
Trainable parameters = Weight + Bias
= (400 x 120) + 120 = 48120

Figure 2.7: Fully Connected Layer

The last layer ends up with a 10-feature output feature map after the fully connected layer.

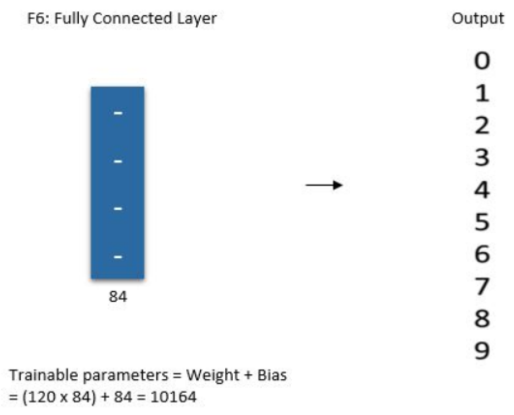

F6: Fully Connected Layer
84
Trainable parameters = Weight + Bias
= (120 x 84) + 84 = 10164
Output
0
1
2
3
4
5
6
7
8
9

Figure 2.8: Fully Connected Output Layer

2.6 Convolution Neural Network with Recommender Systems Textual Information Processing

Apart from image classification, recent study has found that Convolution Neural Network can be used for natural language processing. Convolution Neural Network can be seem as an feature extraction function that is applied to word maps to extract higher level features. Sentiment analysis and question answering are examples of tasks that CNN can operate. In 2008, Collobert and Weston have successfully applied the Convolution Neural Network architecture to natural language processing tasks such as part-of-speech tags, chunks etc.[13]. The results demonstrate that the performance of natural language processing tasks processed by Convolution Neural Network has improved. In 2014, Yoon Kim [1] published a variant architecture of Collobert 's model[13] and the model with only one convolution layer for the training performs well. The basic structure of Kim's model is as follows:

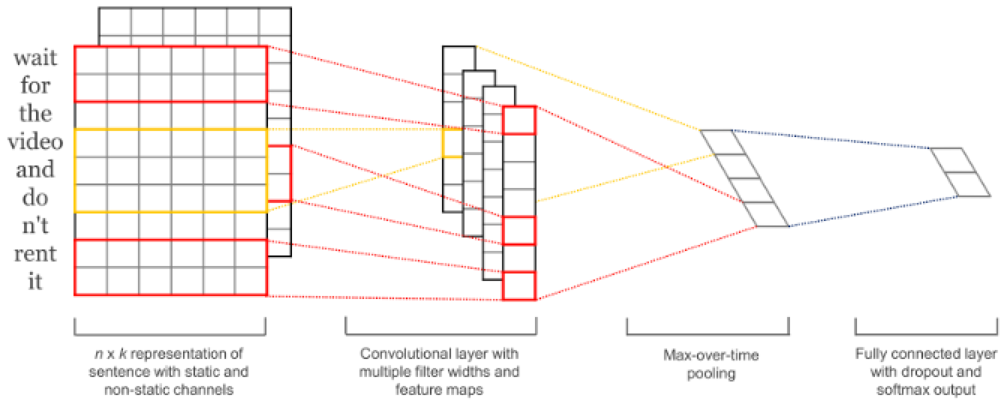

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 2.9: One Convolution layer architecture[1]

In this work,each sentence are tokenized into words before the processing, which are later transferred into a word embedding matrix. The height of the input matrix represents the length of the sentences. The difference of the convolution layer filter between this model and LaNet-5 model is the width of the filters. In this model, the width of filter is equal to the width of the input matrix, which is k. Then the max-pooling layer is applied to the output feature map to capture the most important features from each map.

Thus, Convolution Neural Network can be applied to the recommender systems as well. Donghyun Kim et al. proposed a novel context-ware Convolutional matrix factorization model that integrates Convolution Neural Network into probabilistic matrix factorization (ConvMF)[14]. He used the dataset obtained from MovieLens and Amazon. The textual item descriptions were retrieved by related items from IMDB. Those textual information was processed by using Convolution Neural Network. Matrix factorization technique was also used to decompose the sparse user-item matrix to solve the data sparsity problem within this paper. The results shows that this model performed well and improved the rating prediction accuracy.

Our experiment is based on the experiment of this paper, the differences is that we are using different datset. The above paper trained the model with the textual movie descriptions obtained from IMDB, while I am using the textual user demographic information. But both textual dataset are needed to be trained with Convolution Neural Network. According to the differences of dataset, the architecture would be different as well.

2.7 Conclusion

Recommender systems as the tools of filtering information from high volume of commercial data. The overall recommendation process for a recommender system can be split into two steps:

- collect all user preferences information and item information,
- and generate personalized recommendations from those items to users.

State-of-art algorithms have been proposed to provide a high recommendation accuracy for recommender systems, such Collaborative Filtering, Matrix Factorization, Deep Neural Network etc. Memory-based Collaborative Filtering and Content-based Collaborative Filtering are two branches of Collaborative Filtering algorithm [15][16]. For Memory-based Collaborative Filtering method, recommendations are based on preferences of active user's similar users. With finding the similar users who are rating the items in the similar pattern with active user, the predictions to ratings can be generated.Top-k items would be recommended to the active user after sorting those predicted ratings.

According to the data sparsity problem caused with Collaborative Filtering technique, matrix factorization was introduces to solve this problem by reducing the user-item matrix dimension[8]. The doc product by two matrices, user feature matrix and item feature matrix, would be target user-item matrix. Thus, matrix factorization technique are usually combined with Collaborative Filtering algorithm.

Besides, Deep Neural Network was introduced to deal with the textual information to operate the matrix factorization process by Dongyun Kim[1]. The novel context-aware recommendation model performs well and successfully captures small contextual difference of a word in a sentence even the data is extremely sparse.

MovieLens dataset are the most commonly used dataset for recommender systems research purposes. MovieLens one million dataset contains about one million ratings from 6.000 users on about 4,000 movies. There are three data files within this dataset, ratings.dat, users.dat and movies.dat.

Few papers test with the user demographic information for the recommender system performance, such as one paper published by Laila Safoury and Akram Salah [17] solving the cold-start problem by using Collaborative Filtering technique. Most researches about recommender systems are only use ratings dataset that only contains users previous taste information, neglecting the user demographic data file.

Thus, my initiative of this thesis is to implement a recommender system with user demographic information and test if the performance of recommender systems would be improved by using this dataset. The baseline algorithm would be the combination of Collaborative Filtering algorithm and the matrix factorization technique. Convolution Neural Network technique will be used to process some textual information within the user demographic information. The expected mean square error result implemented with user demographic information, such as age, gender, and occupation will be lower than that experiment without user demographic information. Detailed explanations of this novel architecture would be discussed in Chapter 3.

Chapter 3

Methodology

In this chapter, dataset description, architecture and code implementation of three experiments will be discussed. Lastly, the model evaluation will be explained.

3.1 Datasets Description

The implementation of this dissertation was based on the MovieLen 1-million datasets. MovieLens datasets are collected from a web-based movie recommender systems called MovieLens and are maintained by GroupLens Research[18].

There are three datasets: ratings.dat, users.dat, and movies.dat. MovieLen 1-million datasets contain approximately 1,000,000 ratings from about 6,000 users on around 4,000 movies. Table 4.4 indicates the data format within three files. Ratings.dat contains ratings from users. Each user has rated at least 20 movies and each rating is in a 1-to-5 scale. The rating with 5 means good film and 5 means bad film. Users.dat contains user demographic information such as age, gender and occupation. Movies.dat saved movie titles and movie genres for each movie.

File name	Data Format
ratings.dat	UserID::MovieID::Rating::Timestamp
users.dat	UserID::Gender::Age::Occupation::Zip-code
movies.dat	MovieID::Title::Genres

Table 3.1: Data Format

3.1.1 Datasets Analysis

According to my research of recommender systems papers, almost all papers are experimented only based on ratings dataset. In order to confirm my hypothesis that users preferences would be different according to different age, gender and occupation, I did some further analysis.

Analysis with Different Gender Groups

After merging all three datasets, I group the complete dataset by different genders. For each group, calculate the movie numbers with ratings more than 3 according to different genres. The results shows that different gender groups do have different preferences towards movie genres.

- The top-five preferences for females are

  [Drama(22%),Comedy(14%),Romance(12%),Action(10%),Thriller(9%)];

- while for males are

  [Drama(20%),Comedy(12%),Action(12%),Thriller(10%),Romance(9%)],

Analysis with Different Age Groups

The top-five preferences for users with different ages are different:

- for users with the age 21-30:

  [Drama(19%),Comedy(13%),Action(12%),Thriller(10%),Romance(9%)]

- for users with the age 31-40:

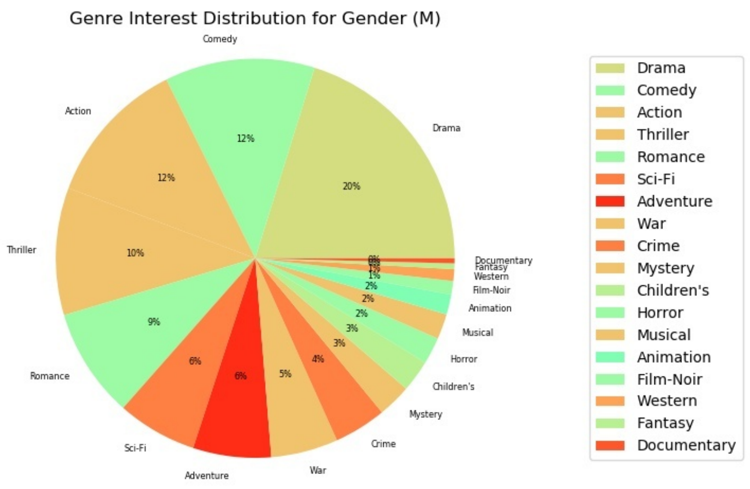
  [Drama(21%),Comedy(13%),Action(11%),Romance(10%),Thriller(9%)];

- for users with the age 41-50:

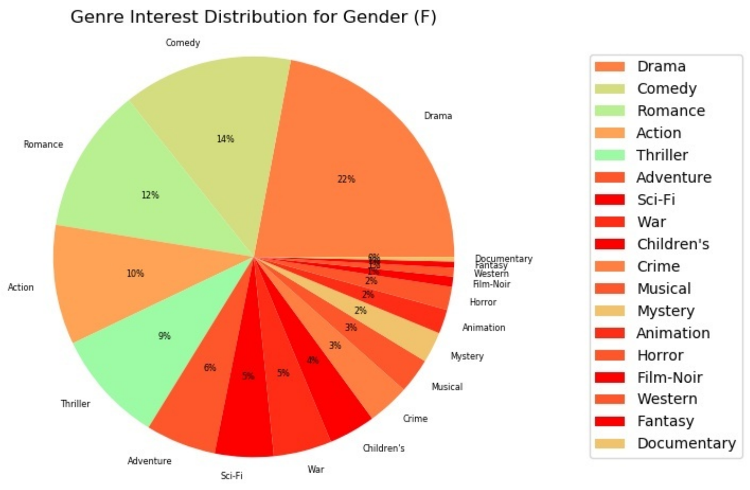
  [Drama(23%),Comedy(12%),Action(10%),Romance(10%),Thriller(10%)];

- for users with the age 51-60:

  [Drama(25%),Comedy(11%),Thriller(10%),Romance(9%),Action(9%)].

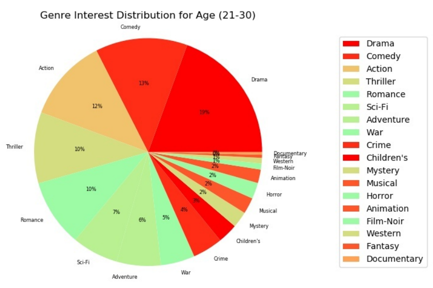


(a) Genre Interest Distribution with Males

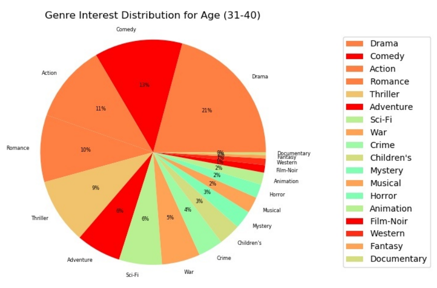


(b) Genre Interest Distribution with Females

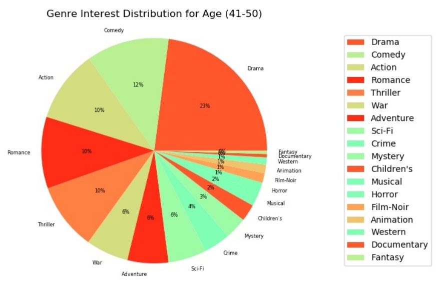
Figure 3.1: Genre Interest Distribution of Different Genders



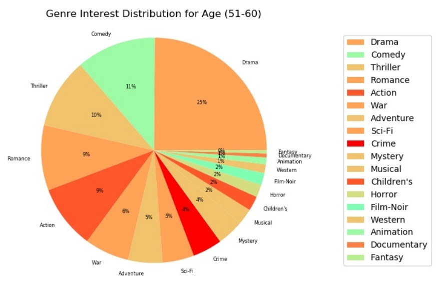
(a) age 21-30



(b) age 31-40



(c) age 41-50



(d) age 51-60

Figure 3.2: Genre Interest Distribution of Different Ages

Analysis with Different Occupation Groups

The similar results are given by the data grouped by different occupations. Users with different occupations have different movie preferences as well. The top-five preferences for users

- with the occupation administrator:

  [Drama(22%),Comedy(13%),Action(11%),Romance(10%),Thriller(10%)];

- with the occupation artist:

  [Drama(21%),Comedy(12%),Romance(10%),Action(10%),Thriller(9%)];

- with the occupation doctor:

  [Drama(23%),Comedy(14%),Romance(12%),Thriller(12%),Action(9%)];

- with the occupation educator:

  [Drama(24%),Comedy(13%),Romance(11%),Action(9%),Thriller(9%)];

- with the occupation engineer:

  [Drama(19%),Comedy(13%),Action(12%),Thriller(9%),Romance(9%)];
- with the occupation entertainment:

  [Drama(20%),Thriller(12%),Comedy(12%),Action(11%),Romance(8%)];
- with the occupation executive:

  [Drama(21%),Action(12%),Thriller(12%),Comedy(11%),Romance(8%)];
- with the occupation healthcare:

  [Drama(23%),Action(12%),Thriller(11%),Comedy(10%),Romance(9%)];
- with the occupation homemaker:

  [Drama(18%),Thriller(15%),Action(14%),Romance(11%),Comedy(9%)];
- with the occupation lawyer:

  [Drama(20%),Comedy(14%),Romance(10%),Action(10%),Thriller(9%)];
- with the occupation librarian:

  [Drama(27%),Comedy(14%),Romance(12%),Thriller(9%),Action(8%)];
- with the occupation marketing:

  [Drama(22%),Action(11%),Comedy(11%),Thriller(10%),Romance(9%)].

3.1.2 Conclusion

Overall, all above results show that user demographic information such as age, gender, occupation are contributing to a user's taste for items. In our cases, that is movies. Thus, if we can use the user demographic information during the recommendation process, I believe the final recommendations will be more personalized. As a result, our recommender systems performance would be improved.

3.2 Data Preparation

3.2.1 Users Dataset

Attribute Name	Data type	Value Ranges
Age	Categorical	1-56
Gender	Character	M,F
Occupation	Categorical	0-20

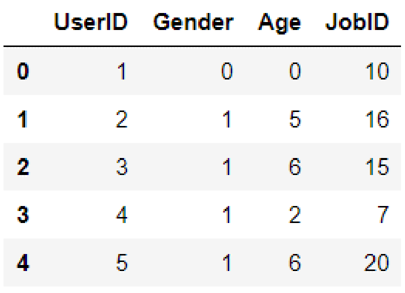
Table 3.2: Attribute Types (users.dat)

Table 3.2 contains the original attribute types for all user demographic information. Age indexes are transferred with continuous numbers 0-6 as shown in Table 3.3.

Original Age Index	Ranges	Final Age Index
1	Under 18	0
18	18-24	1
25	25-34	2
35	35-44	3
45	45-49	4
50	50-55	5
56	56+	6

Table 3.3: Age ranges

22 occupations were changed to continuous numbers from 0-21 as shown in Table 3.4: Gender information 'F' and 'M' are transferred to 0 and 1.



	UserID	Gender	Age	JobID
0	1	0	0	10
1	2	1	5	16
2	3	1	6	15
3	4	1	2	7
4	5	1	6	20

Figure 3.3: users.dat

Occupation Index	Occupations
0	other or not specified
1	academic/educator
2	artist
3	clerical/admin
4	college/grad student
5	customer service
6	doctor/health care
7	executive/managerial
8	farmer
9	homemaker
10	K-12 student
11	lawyer
12	programmer
13	retired
14	sales/marketing
15	scientist
16	self-employed
17	technician/engineer
18	tradesman/craftsman
19	unemployed
20	writer

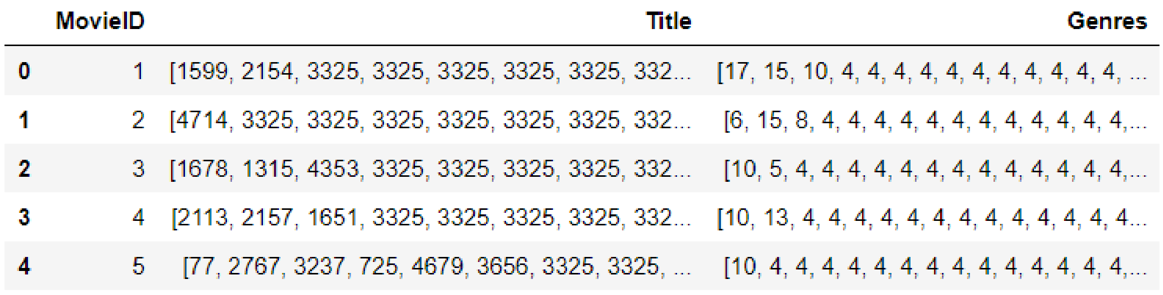
Table 3.4: Occupations

3.2.2 Movies Dataset

Movie genres are textual data and should be transferred into continuous categorical data with the range 0 to 17. Each movie belongs to one or more than one genres, thus the genre attribute for each movie would be a list of genres index.

As for movie titles, each movie title has different length of words which will be complex when generating the input training matrix. In this context, the movie titles are defined with the same length of 18 and using 'PAD' to fill blank positions.

The processed movies dataset structure is in Figure 3.4:



	MovieID	Title	Genres
0	1	[1599, 2154, 3325, 3325, 3325, 3325, 3325, 332...	[17, 15, 10, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, ...
1	2	[4714, 3325, 3325, 3325, 3325, 3325, 3325, 332...	[6, 15, 8, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,...
2	3	[1678, 1315, 4353, 3325, 3325, 3325, 3325, 332...	[10, 5, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,...
3	4	[2113, 2157, 1651, 3325, 3325, 3325, 3325, 332...	[10, 13, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4...
4	5	[77, 2767, 3237, 725, 4679, 3656, 3325, 3325, ...	[10, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4,...

Figure 3.4: movies.dat

Those numbers would the the index of all textual information.

3.3 Experiment 1: Implementing Collaborative Filtering Matrix Factorization Model

Collaborative Filtering Matrix Factorization model is a variant based on the model publish by Yehuda Koren et al.[8] in 2009. The idea of matrix factorization is to solve the data sparsity problem within the collaborative filtering algorithm. The sparse user-item matrix will result in the low-accuracy when finding similar users. In addition, the final recommendation performance would be influenced.

The dimension of ratings matrix can be reduced into two smaller feature matrices p and q. The goal of matrix factorization is using gradient descent method to update p and q and find the find optimal results with the lowest error. The Collaborative Filtering Matrix Factorization model is used in this dissertation to train the dataset with ratings dataset and without user demographic information. The overall process is divided into three steps: data splitting, model training and model evaluation.

3.3.1 Data Splitting

The input dataset is the MovieLens 1-million dataset and the ratio of training set with testing set is set to 0.8. Instead of shuffling the complete dataset and split data into training set and testing set directly, dataset was split according to users. Each user has rated at least 20 movies. For each user in the dataset, split 0.8 rating records to training set and rest 0.2 rating records are added into testing set. This could make sure that each user can be tested with same ratio of ratings.

3.3.2 Model Training

After splitting the dataset, all the training data can be used to train the model. We denote p as user feature matrix while q is represented for the movie feature matrix. The doc product of these two matrices are the predicted ratings.

$$\hat{r}_{u,i} = q^T q \tag{3.1}$$

In order to solve matrix p and matrix q, we can define an optimization problem to minimize the mean square error between the predicted ratings by our model and the true ratings. Mean square error can be used in such stepwise regression models to determine the numbers of predictors included in the model by the given observations.

The objective function can be represented as:

$$L = argmin_{p^*,q^*} \sum_{(u,i)\in K} (r_{ui} - (q_i)^T p_u)^2 + \lambda(||q_i||^2 + ||p^u||^2) \tag{3.2}$$

The right side term that starts with λ is a regularization term to prevent the overfit of our model. This optimization problem can be solved by using gradient descent method. Take the partial derivative step by step respect to p and q to find the optimal values. The iteration steps can be changed according to different scenarios.The input of this model is:

- the number of features,

- the learning rate γ,
- regularization term λ,
- iteration steps,
- and the ratings dataset.

In my experiment, I have tested the performance with the feature number 10,20,50,100 and 200. The max iteration is set to 50 and 100.

Python pseudocode for the training is as follows:

```
Initialize a random user feature matrix p and the item feature matrix q.
for step=1,2,3,...100 do:
    for (user, movie, rating) in dataset_R:
        calculate $\hat{p} = (q_{movie})^T \times p_{user}$
        error=$rating - \hat{p}$
        update $p_{movie}$ and $q_{user}$
        objective function ($q_{movie}$,$p_{user}$,rating)
    end of for loop
end of for loop
```

3.3.3 Model Evaluation

The Collaborative Filtering Matrix Factorization model can be evaluated by the most common algorithm, mean square error. Because other evaluation matrices, such as precision, recall or accuracy will not calculate the exact differences between the predicted rating figures and the the true ratings but only count the how many movies are predicted correctly. Thus, mean square error is chosen for comparing the statistical models to explain how well the model is. The mean square error is computed by the mean $(\frac{1}{k}\sum_{i=1}^{k})$ of the squares of errors between predicted values and true values $(\hat{r_i} - r_i)^2$ for testing dataset.

In our model, mean square error is used both in the training part and the evaluation part. For the training part, the model will trained by optimize the errors of training set to find the user feature matrix and the item feature matrix. As for the evaluation part, the mean square errors will be calculated by using testing set and the trained matrix p and q. The dot product of user feature matrix p and item feature matrix q will be the user-item matrix we used for the recommendation.

The evaluation part is followed after the model training for each iteration. Once we get the user-item matrix, the evaluation process in Python pseudocode will be as follows:

```
Input: user feature matrix p, movie feature matrix q, test set, recommendation numbers(20)
1) Calculate user-item matrix by using p and q:
for (user, movie, rating) in test dataset:
  $\hat{rating} = p_{user} \cdot q_{movie}^T$
2) Generate top-20 items for each user in test set. For each user in the test set, sort the items according to the predicted ratings in the user-item matrix. For each item that each user have not rated, recommend to that user. The recommendation number in this experiment is 20.
for user1 in test datset:
  for item1 in sorted(items):
    if user1 have not rate for item1:
      errors += $(ratings(user1, item) - true(user1, item))^2$
  mean of errors= errors/total error numbers
```

There are 200 iterations for the Experiment 1, thus the total mean square errors will be 200. Detailed results will be shown in Chapter 4.

3.3.4 Parameter Tuning

As the input of model training has five parameters, we can compare the model performance by changing the input parameter values. Feature numbers and the max iteration steps are two parameters that would affect largely to the model. Limiting one of the parameters and changing another one would give different insights of the model.

During my training, the feature number has been set from 10, 20,50,100 and 200 while controlling the iterations to be the same. The max iterations are set to 50 and 100 while set the feature numbers to be the same. These two parameters are affecting the mean square error in the similar way. With the increasing of feature numbers, the mean square error is reducing. However, there is a boundary for the feature numbers.

The mean square error would not continuously reducing when stepping out of that boundary. Same as to the iteration numbers, before the limitation boundary, with the increasing of iteration numbers, the mean square error will reduce. More details of the results will be discussed in Chapter 4.

3.4 Experiment 2: Implementing Convolution Neural Network Collaborative Filtering Matrix Factorization Model

The overall architecture of Convolution Neural Network Collaborative Filtering Matrix Factorization trained with MovieLens user demographic information is based on Kim Yoon's work published in 2004[1]. Kim used only one convolution layer to train the model for textual information and successfully complete the sentence classification task. With two channel of 9×6 input matrices, the outout is a two-element matrix indicating whether the sentence is positive or non-positive. Kim's architecture is shown in Figure 3.5.

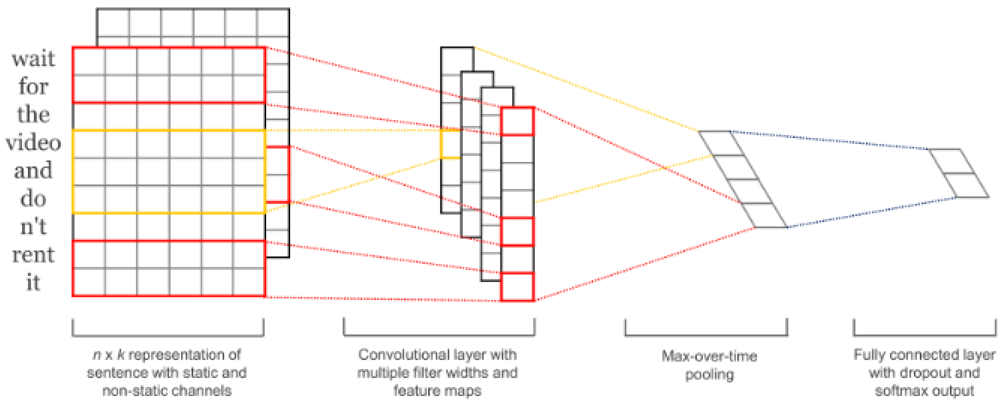

wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 3.5: One Convolution layer architecture[1]

Figure 3.5 shows the architecture of the novel Convolution Neural Network Collaborative Filtering Matrix Factorization model in this thesis. Tensorflow and Keras packages were used in this experiment. It is implemented by two parts, the implementation of user information and the implementation of movie information. The model is trained with different epochs and each epoch is trained with all MovieLen's one million dataset. Within each epochs, 80% of one million dataset are splited into training set and the rest 20% belong to the test set. The 80% training set later are divided into different batches and are trained separately. Each batch contains 256 data. To control the iteration number same as the Experiment 1, 5 and 100 iterations have been trained.

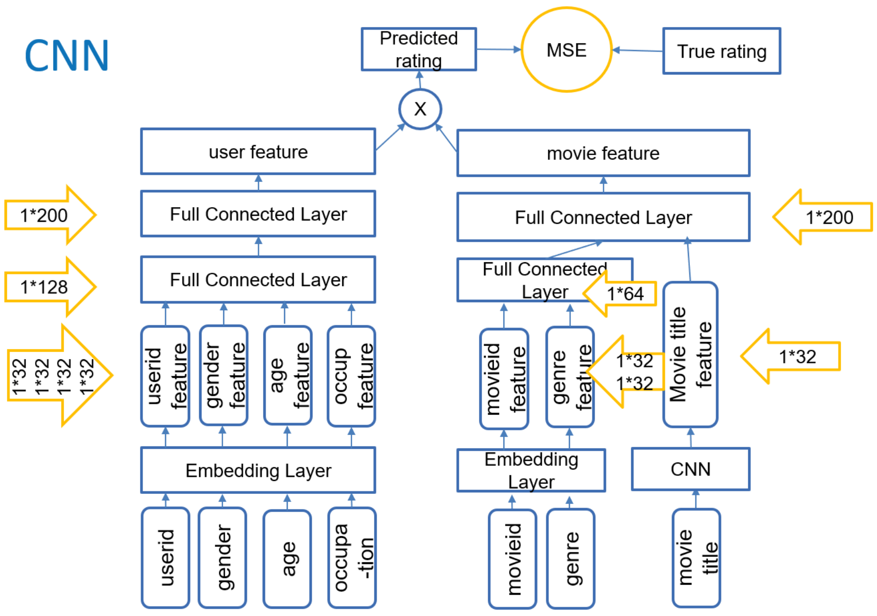


Figure 3.6: Convolution Neural Network Matrix Factorization Architecture

3.4.1 Implementation of user information

The implementation process of user information is on the left side of Figure 3.5, from bottom to the top.

- 1. Get user embedding matrix:

  For each input of userid, user age, user gender, user occupation, four 1×32 feature embedding matrices can be generated by tf.nn.embedding_lookup() in Tensorflow packge.

- 2. Trained with fully connected layer:

Four 1×32 embedding matrices can be fully connected to a 1×128 feature matrix. Figure 3.6 shows the complete process of full connection. Embedding matrices that passed through full connected layer, all features from each embedding matrix will be captured and added to another feature matrix. Therefore, the output matrix after the first full connected layer would contain all features of userid, user age, user gender and occupation.

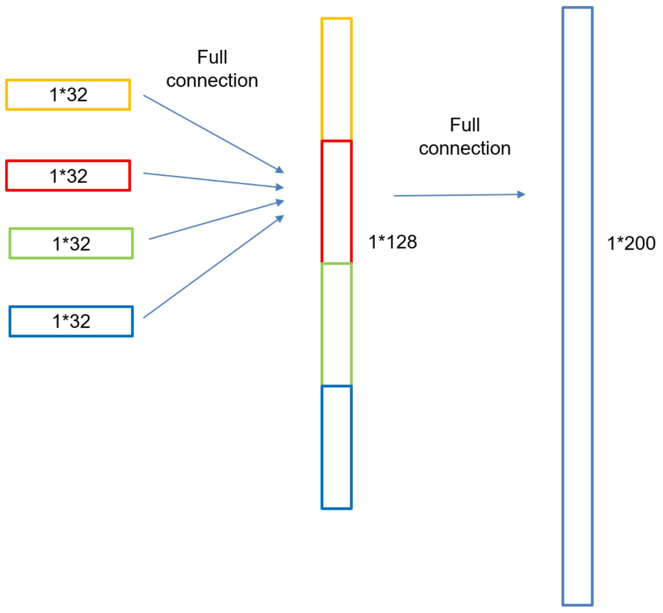


Figure 3.7: Full Connected Layer

- 3. Trained with the second full connected layer:

  The idea of this layer is to expand those captured features to a bigger feature matrix. As the results, the last 1×200 feature matrix would capture enough features from users. The output matrix is the user feature matrix p that we are going to find. We can also change 200 features to be any other numbers, because this is one parameter we can decide within the experiment indicating how many features we want to extract from users.

  In Experiment 2, the feature number was set to 128 and 200.

3.4.2 Implementation of movie information

The implementation of movie information is similar to the above process of user information.

- 1. Get movie embedding matrix:

  For a input of movieid, one 1×32 feature embedding matrix can be generated. Movie genres is slightly different due to the fact that each movie may belong to one or more genres. Each genre is a 1×32 feature embedding matrix. Therefore, all genre matrices for one movie should be added together according to features and to be calculated as one 1×32 genre feature matrix. Thus, for movies belonged to more than one genres, there would be more than one 1×32 embedding matrix. After adding all matrices, the final matrix would be one 1×32 movie genre embedding matrix we are looking for. This process can be easily realized by reducing one dimension for all genre embedding matrices by tf.reduce_sum() method in Tensorflow package.

- 2. Get movie title feature matrix by using Convolution Neural Network:

  Getting movie title feature matrix is the key part of the overall architecture. It is a one-convolution-layer operation but operated with four filter sizes. Movie titles can be treated as sentence with the sentence length 15. The differences with Kim's architecture are the input matrix size, the convolution layers number and the filter number, epoch numbers, batch numbers. The results show that the modified architecture works better with my training datset.

  The architecture for this thesis can be found in Figure 3.8. In this experiment, the input matrix size is 15×32 because the length of each movie title is 15 and each word with each movie title contains 32 features. The filter size is 2,3,4 and 5, which means 2×32, 3×32, 4×32, and 5×32 filters were used to slid across the height of input matrix. The filter number is set to 8. With four different filter sizes, the input matrix would be filtered four times by using those four different filters. Thus, after each convolution layer, the output matrix would be eight $(15 - filtersize + 1) \times 1$ feature matrix.

  Next process is passing all those feature matrix through a max-polling layer with filter size equal to $15 - filtersize + 1$. The purpose of this layer is to find the most important features among all features to prevent overfit of the model. In addition, it reduces computational cost by reducing numbers of features to learn.

The final process is through the full connected layer and droupout process. Dropout layer means to randomly set input elements to zero with a given probability. In this thesis, the droupout ratio is set to 0.2. The dropout ratio for Kim's architecture is 0.2. Thus, I am using the same ratio as him. In the future, all those parameters will be re-trained and to find a best combination for this model.

The final output is a 1×32 feature matrix.

Take an example of 2×32 convolution layer. With the 15×32 input matrix, after the first convolution layer with eight 2×32 filters, the output feature matrix would be eight 144×1 feature matrix. Then, after passing the 144×1 feature matrix to max-pooling layer, the output would be a 1×8 matrix. Fully connecting four 1×8 matrix would generate a 1×32 feature matrix.

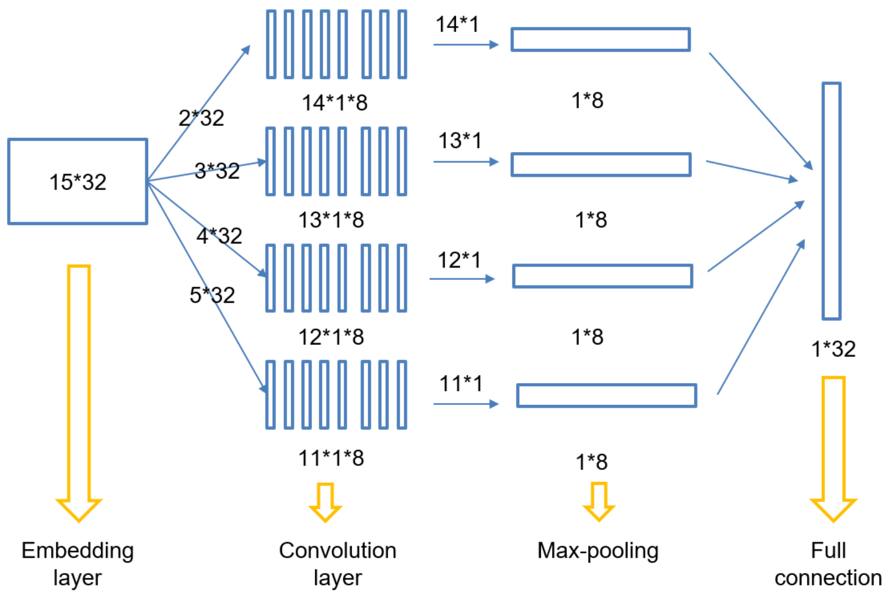


Figure 3.8: Movie Title Convolution Neural Network Architecture

- 3. Process through the fully connected layer:

All three feature matrices generated from movieid, movie genres, and movie titles can then be fully connected to a 1×200 feature matrix. That is the movie feature matrix q that we are looking for.

The feature number of the user feature matrix and the item feature matrix should be the same. Because in the user feature matrix, each row is described by the users interest in different features. In the item feature matrix, each column is the item described by the features.

As Experiment 2 was based on the Kim's [1] architecture. Thus, in Experiment 2, the item feature number was set to 200, same dimension as Kim's one. The iterations of the test comparing with Kim's model were set to 5. But other parameters, such as filters size, filters number, batch size, drop out rate were modified ones. The modified model is proved to have higher performance than Kim's model. Detailed results will be discussed in Chapter 4.

The Python pseudocode for the training is as follows:

```
Input user information and movie information.
for epoch=1,2,3,...100 do:
  train_batch, text_batch = get_batches()
  for batch_i in (len(trainset)// batch_size):
    get user feature matrix p and item feature matrix q
    calculate $\hat{p} = (q_{movie})^T \times p_{user}$
    error=$(rating - \hat{p})^2$
  avg_error= error/total batch numbers
  sum_error +=avg_error
  end of for loop
errors = sum_error/ total epoch number
update p and q
end of for loop
```

3.4.3 Generating predictions of ratings

Our final aim for this experiment is to find user feature matrix and movie feature matrix and use their doc products to generate predicted ratings to fill in missing values within user-item matrix. The doc product of user feature matrix and movie feature matrix are the predicted rating we need according to the idea of matrix factorization.

The recommendation process is similar as what we did in Experiment 1. Once we fill the user-item matrix, user similarity matrix can be calculated by the consine similarity matrix. The top-20 recommendation will generated to each user in test dataset.

Python pseudocode for the evaluation is as follows:

Input: user feature matrix p, movie feature matrix q, test set, recommendation numbers(20)

```
1) Calculate user-item matrix by using p and q:
for (user, movie, rating) in test dataset:
    $\hat{rating} = p_{user} \cdot q_{movie}^T$
```

2) Generate top-20 items for each user in test set. For each user in the test set, sort the items according to the predicted ratings in the user-item matrix. For each item that each user have not rated, recommend to that user. The recommendation number in this experiment is 20.

```
for user1 in test datset:
    for item1 in sorted(items):
        if user1 have not rate for item1:
            errors += $(ratings(user1, item) - true(user1, item))^2$
    mean of errors= errors/total error numbers
```

There are 100 iterations for the Experiment 2, thus the total mean square errors will be 100. Detailed results will be shown in Chapter 4.

3.4.4 Model Evaluation

In order to better compare the experiment without user demographic information and the experiment with user demographic information, we are using the same evaluation measurement, mean square error.

In this experiment, mean square error are calculated at each iterations (epochs). There are 100 epochs within Experiment 2, each epoch contains 256 batch training data. The mean error for all batches is the error of one iteration. Thus, there are 100 error numbers as Experiment 1.

3.4.5 Trained with different Parameters

Because of the complexity of model in Experiment 2, there are numbers of parameters that we can change for different tests. Parameters such as the embedding layer size (32), movie title length(15), filter sizes(2,3,4,5), filter numbers(8), epoch numbers(8), batch sizes(256),feature numbers(200), dropout ratio(0.2) etc. However, the purpose of this work is to test that using the user demographic information would improve the recommder systems performance. Building the best Convolution Neural Network architecture for recommender systems will be work on in the future.So far, we only focus on feature numbers and iterations as what we focused on Experiment 1. Other parameters is set ans slightly modified according to Kim's architecture[1]. 128 features and 200 features were trained with the model. 5 and 100 iterations are compared with Experiment 1. Detailed results will be discussed in Chapter 4.

3.5 Experiment 3: Implementing Collaborative Filtering Model with User Demographic Information and without User Demographic Information

The purpose of this added experiment is to prove that the improvement performance of Experiment 2 was resulted from the user demographic dataset instead of the Convolution Neural Network technique. Experiment 3 actually contains two small tests. One test is using the user demographic information dataset and another one without using the user demographic information. The reason that I have not process another similar experiment as Experiment 2 to train the dataset without user demographic information is that, Convolution Neural Network is used in Experiment 2 to train the textual information within the user demographic information. However, the dataset without user demographic information does not contain any textual data, there is no way to do another experiment for dataset without user demographic information with Convolution Neural Network.

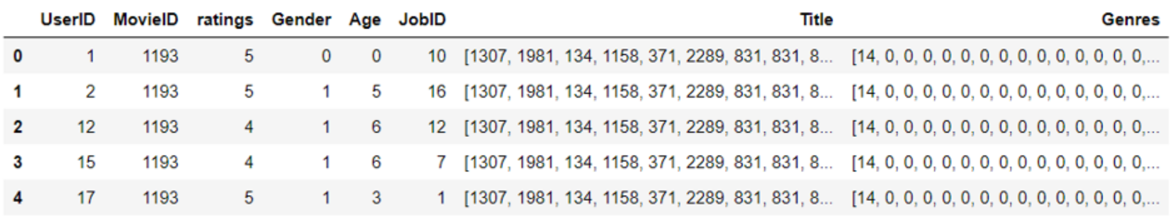
The idea of Experiment 3 is to train two dataset with User-Based Collaborative Filtering algorithm only. Once the user-item matrix is calculated, no matrix factorization will be used to fill in those missing numbers. All missing ratings will be calculated by each users' similar users. As there is no optimization problem, the experiment will be only processed once. Thus the evaluation will be only one number. The evaluation matrix will be different according to different ways of implementation. Accuracy, recall and precision will be used in this context. The two evaluation results for Experiment 3 are only compared within Experiment 3 to prove the influence of user demographic information dataset.

User-based Collaborative Filtering technique is to locate the similar users with the active user(the user whom the predictions are for) and use the similar users preferences to recommend items to the active user. In order to find the similar users, user-item matrix are measured by user similarity matrix. In all three experiments, cosine similarity matrix was chosen for the measurement.

The overall process for Experiment 3 is similar as Experiment 1, separated to three steps: data splitting, model training and model evaluation.

3.5.1 Data Splitting

The merged datset is shown in Figure 3.9 is the dataset with user demographic information, the same dataset used as Experiment 2. Another dataset without user demographic information is the same as the dataset used in Experiment 1. For each user in the dataset, split 80% rating records totraining set and rest 20% rating records are added into testing set. The training dataset was split by users to make that each user can be both tested and trained within the model.



	UserID	MovieID	ratings	Gender	Age	JobID	Title	Genres
0	1	1193	5	0	0	10	[1307, 1981, 134, 1158, 371, 2289, 831, 831, 8...	[14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
1	2	1193	5	1	5	16	[1307, 1981, 134, 1158, 371, 2289, 831, 831, 8...	[14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
2	12	1193	4	1	6	12	[1307, 1981, 134, 1158, 371, 2289, 831, 831, 8...	[14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
3	15	1193	4	1	6	7	[1307, 1981, 134, 1158, 371, 2289, 831, 831, 8...	[14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
4	17	1193	5	1	3	1	[1307, 1981, 134, 1158, 371, 2289, 831, 831, 8...	[14, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...

Figure 3.9: Merged dataset

3.5.2 Model Training

The Experiment 3 is using User-based Collaborative Filtering algorithm for the model training. The user-item matrix used for the recommendations is calculated by the similarity users' preferences.

User-based Collaborative Filtering algorithm is operated as three steps:

- Step 1: Look for users who share the same rating patterns with the active user (the user whom the prediction is for). There can be one or two or even thousands similar users, but each one have the different similarity factor according to there rating patterns. The similarity matrix within users is calculated by consine similarity matrix.
- Step 2: Use the ratings from those similar users found in step 1 to calculate a prediction for the active user predict ratings for unrated items
- Step 3: Recommend top-n item from those filled items. In my experiment, 20 movies will be recommended to each user. In this experiment, the recommendation number is 20, same as the above two experiments.

3.5.3 Model Evaluation

Once we generate the Top-20 movies for each user in the test set, accuracy, recall and precision can be calculated by comparing the predicted items with the true rated items.

Python pseudocode for the Experiment 3 is as follows:

```
Input: user feature matrix p, movie feature matrix q, test set, recommendation numbers(20)
1) Calculate user-item matrix by using p and q:
for (user, movie, rating) in test dataset:
    $\hat{rating} = p_{user} \cdot q_{movie}^T$
2) Calculate user similarity matrix by calculating cosine similarity matrix between user vectors.
for user1 in test user:
    for user2 in test user:
        similarity factor (user1, user2)= cosine(user1,user2)
```

3) Generate recommendations for users. Traverse all users in the test set, predict ratings that each user will give to each item according to his similar users preferences.

```
for user1 in test datset:
  for user2 in similar users:
    for item in items:
      if user2 have not rate for item:
        ratings(user1,item) = similarity factor(user1, user2) * ratings(user2, item)
```

The precision is the proportions of the correct predictions and the all predictions. That means how many predictions are predicted correctly.

The recall is the proportion of positive predictions and all positive observations. That is used to calculated how many items that users are truly rated are predicted by the system.

The accuracy is similar to precision, the number of true positive and true negative predictions divided by total predictions.

Detailed results will be discussed in Chapter 4.

Chapter 4

Results and Discussion

This Chapter will list all the results compiled by two recommender system model. The explanation will be structured with three parts, results for three experiments and the comparison of those results.

The evaluation matrix for Experiment 1 and Experiment 2 is measured by the mean square error(mse). Calculating the mean of the squared errors for 100 iteration between the predicted ratings with the true ratings will better capture how well the model is.

The evaluation matrix for two small tests within Experiment 3 is measured by the precision, indicating how many predictions are correctly predicted by the system.

The results for Experiment 1 and 2 are two comparative results to test if the recommender system performance would be improved by the user demographic dataset, while the results within Experiment 3 are the results compared to prove the improvements in Experiment 2 are resulted from the dataset. Therefore, in order to better understand the two process, two evaluation measurements were used in case the mismatch of two results.

4.1 Mean Square Error Results for Experiment 1

The first experiment was using the datset without user demographic information and used the baseline algorithm (Collaborative Filtering algorithm and matrix factorization technique) for the training.

The evaluation of the model is measured by the mean square error. With all the input parameters(the number of features, the learning rate, regularization term, max iteration steps etc.), the number of feature numbers that are going to generated and the max iteration steps to be trained are two parameters that Experiment 1 and 2 focus on.

Three iterations(50, 100) and four different feature numbers (10,20,50,200) are generated had been tested for Experiment 1. 1. The first test is to train the dataset with 50 iterations and generate 10, 20, 50, and 200 features. The mse results for the dataset can be seen in Figure 4.1.

4.1.1 Test 1: Train the dataset with different feature numbers

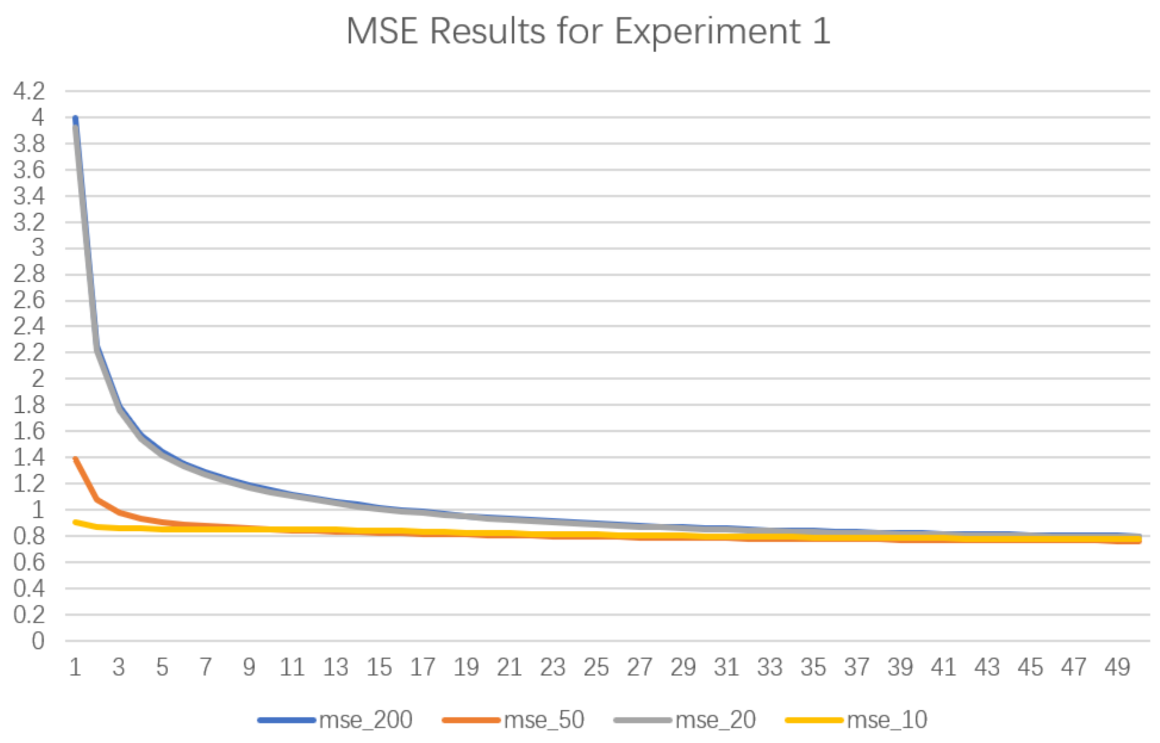


Figure 4.1: MSE results for Experiment 1 trained with 50 iterations

Different colors within Figure 4.1 means different numbers of features that were finally trained. In this experiment, as matrix factorization was used to trained the user features and the item features. In the user feature matrix, each row is described by the users interest in features F1, F2 etc. In the item feature matrix, each column is the item described by the features F1, F2 etc. The more the features are, the better description can be extracted for users and items. As result, their dot product, the user-item matrix will provide more accurate results when generating recommendations.

We can see from Figure 4.1 that with the increasing of the feature numbers, the mse results is reducing. As it is an optimization problem, the last results of four lines are there best results. Four last mse results is in Table 4.1. It shows that the best result for this test, is getting close to 0.760. There must be a boundary between the feature numbers 50 and 200. Before the boundary, with the increasing of feature numbers, the mse results will reducing. Once cross that boundary, the mse results will incresing. However, our job is not to find this boundary, thus the results so far work fine with this experiment.

	MSE
mse_10	0.774
mse_20	0.770
mse_50	0.763
mse_200	0.781

Table 4.1: mse results for different feature numbers

4.1.2 Test 2: Train the dataset with different iterations

This test was operated with 200 features but trained for different iterations. The reason to choose 200 features is that in Experiment 2, it is more meaningful to extract more features than 128 features. Because after fully connected all four user features, we can get 128(32*4=128) features. If we want to capture more user features, the final extracted feature number must be larger than 128. Besides, our model is based on Kim's[1] model, 200 feature numbers is chosen for his model and performed well. Thus, I trained the model for Experiment 2 with 200 features as well. In order to better compare those two experiments, I control the feature number of Experiment 1 to be 200 as well.

The mse results trained for 200 features with 100 iterations can be seen in Figure 4.2.

As we can see that with the increasing of iteration numbers, the mse results is reducing. Finally, the it is getting close to 0.770. 7.70 will be the best result for

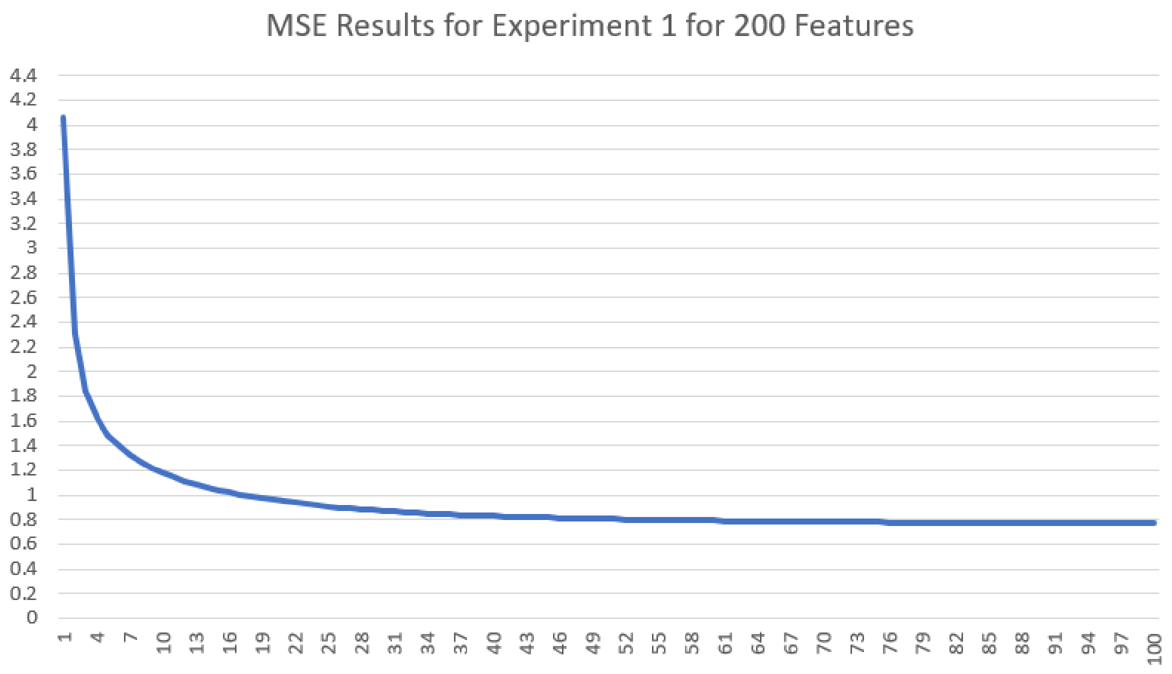


Figure 4.2: MSE Results for Experiment 1 with 200 features

Experiment 1 and will be used to compare the mse result for Experiment 2.

4.2 Mean Square Error Results for Experiment 2

Experiment 2 was trained with the user demographic dataset. The same baseline algorithm was used for the model training as Experiment 1. However, with the complexity of textual information in the user demographic information, state-of-art algorithm, Convolution Neural Network was also used for the matrix factorization implementation. The related parameters for Experiment 2 are the feature numbers, the max iteration numbers, batch sizes etc. But in our experiment, the feature numbers and the max iteration numbers are two influence we focus on.

4.2.1 Test 1: Train the dataset with different parameters

As explained in the Experiment 1, the feature number of Experiment 2 is more meaningful if we extract the feature numbers that is larger than 128. As the model is based on Kim's[1] model. Thus, I operated the tests 200 features and 5 iterations, same setting with Kim's model.

The mse results compared with Kim's model is in Figure 4.3.

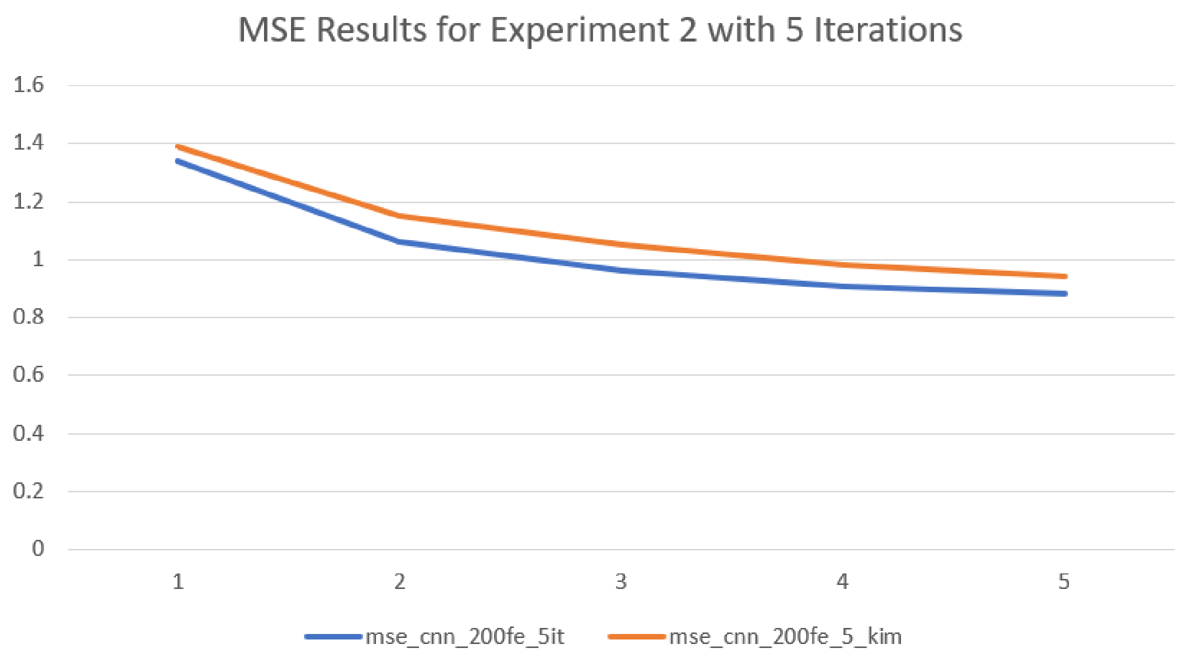


Figure 4.3: MSE Results for Experiment 2 compared with Kim's [1] model

	MSE
mse_Kim	0.941
mse_my model	0.879

Table 4.2: mse results compared with Kim's[1] model

Different colors of lines means different settings of my novel model trained with Convolution Neural Network. Two models are trained with user demographic information dataset. The orange line is the model trained with parameter setting within Kim's[1] paper. The feature numbers is 200. The max iterations is 5. The window sizes are 3, 4, 5. The dropout rate is 0.2. The batch size is 128. The mse result of this model is getting close to 0.941. Another blue line is the model designed by myself with the modified parameters. The mse results of this model is getting close to 0.879.

There are other factors that can affect the final mse results. But if we only compare those parameters, my model is performing around 6.5% better than Kim's model.

4.2.2 Test 2: Train the datset with different feature numbers

This test is to train the user demographic information datset by using my model with different feature numbers. As explained the before, the feature number is better to set to be larger than 128. Thus, I process two tests trained with 128 features and 200 features with 5 iterations. The mse results can be seen in Figure 4.4.

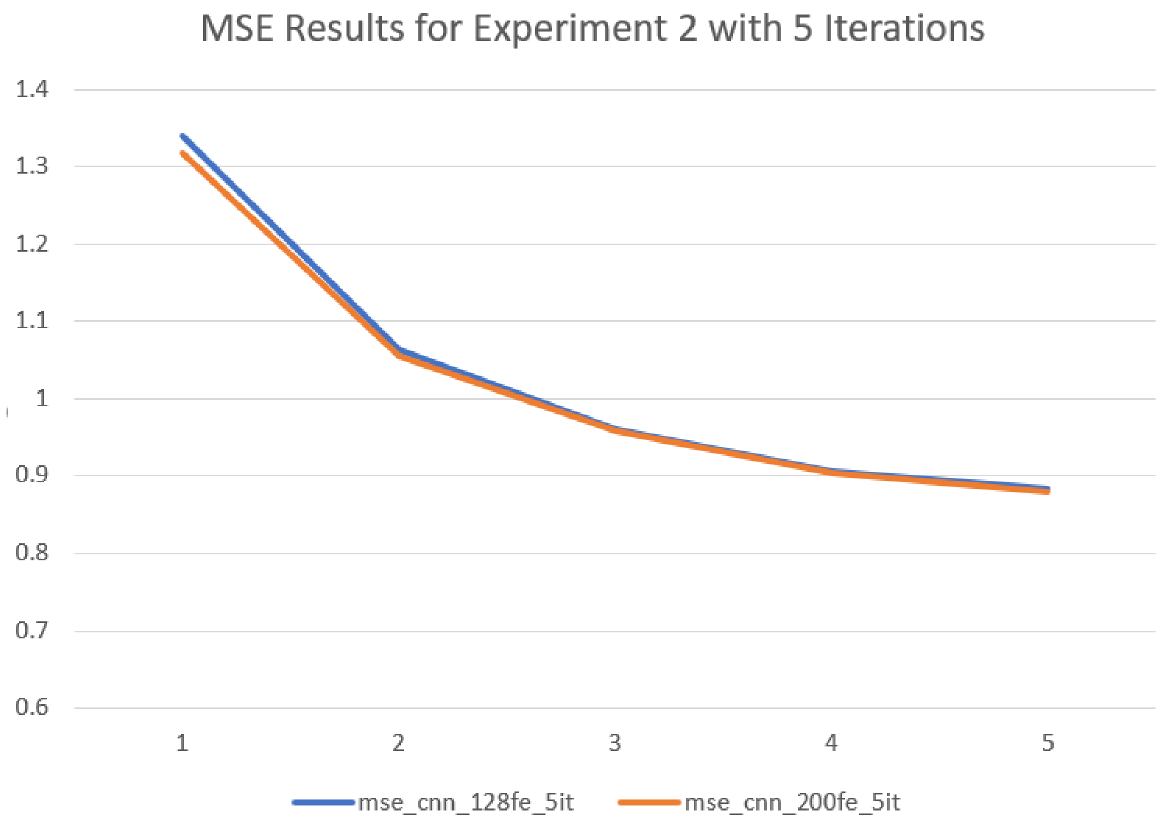


Figure 4.4: MSE Results for Experiment 2 trained with different feature numbers

The Figure 4.4 shows that the mse results trained with 200 features gives better performance than that of trained with 128 features. Even though two results is quite close shown on the figure, the mse results of orange line trained with 200 features(0.879) is lower than that of blue line(0.884).

	MSE
mse_128	0.884
mse_200	0.879

Table 4.3: mse results trained with different feature numbers

4.2.3 Test 3: Train the dataset with different iterations

The last test within Experiment 2 was to train the user demographic information dataset with 100 iterations. The results is shown on Figure 4.5. With the increasing of the max iterations, the best results for this test is getting close to 0.661.

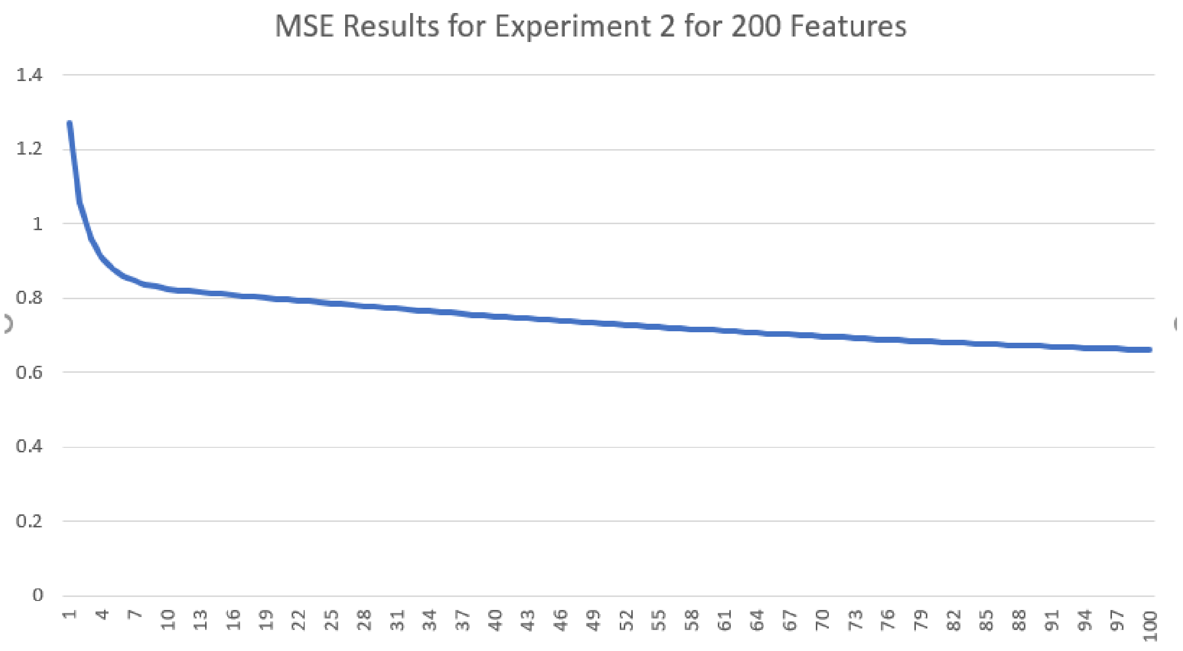


Figure 4.5: MSE Results for Experiment 2 trained with 100 Iterations

4.3 Comparison between Experiment 1 and 2

By comparing the above results, we get that:

- With the increasing of feature numbers, the mse results is reducing.
- With the increasing of iteration numbers, the mse results is reducing as well. The novel modified algorithm trained with user demographic information dataset

performs better than Kim's[1] model trained with user demographic dataset. If we control both experiments with 200 features and 100 iterations, the recommender system's performance is better than that of the experiment trained without user demographic information. Even though. we have controlled most parameters, the algorithms used for two experiments are different as well. The Convolution Neural Network is used to train the textual information within the user demographic information dataset. It is unavoidable to add this technique. Thus, Experiment 3 was added to prove that the improvement in Experiment 2 was resulted from the user demographic dataset.

Experiment number	feature numbers-iterations	MSE
1	10-50	0.774
1	20-50	0.770
1	50-50	0.763
1	200-50	0.781
1	200-100	0.770
2	128-5	0.884
2	200-5	0.879
2	200-100	0.661

Table 4.4: mse results trained with different feature numbers

4.4 Precision for Experiment 3

The aim of Experiment 3 is to confirm that the improvement of Experiment 2 was caused by the user demographic information dataset instead of the Convolution Neural Network algorithm. Thus, the algorithms used within this experiment are same. The only differences between two tests is the dataset. Two tests were processed within this test.

The precision was used to measure two tests. The precision was calculated by the proportion of correctly predicted items among all predictions. That means when the system is generating 20 movies for each user, the evaluation was processed by counting the correctly predicted movies compared to users true ratings divided by the total prediction number.

- Test 1: Train the dataset without user demographic information with User-based Collaborative Filtering Algorithm;
- Test 2: Train the dataset with user demographic information with User-based Collaborative Filtering Algorithm

The precision of test 1 is around 38.5%, while the precision for test 2 is around 46.3%. Thus, the recommender performance trained with user demographic information is around 17% higher than that of trained without user demographic information.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The mse results compared between Experiment 1 and Experiment 2 show that the combination of the user demographic information dataset and the Convolution Neural Network can improve the performance of one recommender system. The mse results for the Experiment 1 trained without user demographic information dataset is around 0.770, while the mse result for Experiment 2 is around 0.661. The Experiment 2 has improved about 15% performance of a recommender system. However, as there are two differences between those two experiments, it is not very clear that it is the user demographic information dataset helps to improve the performance. Thus, Experiment 3 was added to train the recommender system with same algorithm but different datasets.

The final results of Experiment 3 shows that by integrating the user demographic information, the performance of a recommender system is improved. The precision results for the test with user demographic information is around 46.3%, while anothe test trained without user demographic information is around 38.5%. Around 17% improvement has been proved. This added experiment better confirm the results for Experiment 2.

The novel recommender system model built with Convolution Neural Network within Experiment 2 can be used in different recommender system area.

5.2 Future Work

In the future, there will be a lot can be done:

- The first thing I can do is to rebuild a better model with Convolution Neural Network. As we all know, Convolution Neural Network is a very complex model, one little change for one parameter will results in a different result. Thus, parameter tuning should be done in the future to train a better model.
- The second part that can be improved is the Experiment 3. Even though the results have already proved that the user demographic information can improve the performance, the precision is very low. Maybe other algorithms can be chosen in the future to better train the model and give a higher precision.

Bibliography

[1] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1746–1751, Association for Computational Linguistics, Oct. 2014.

[2] A. P. V, "A survey of recommender system types and its classification," *International Journal of Advanced Research in Computer Science*, vol. 8, no. 9, p. 486491, 2017.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, p. 22782324, 1998.

[4] R. Mehta and K. Rana, "A review on matrix factorization techniques in recommender systems," *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*, 2017.

[5] L. Baltrunas and F. Ricci, "Context-based splitting of item ratings in collaborative filtering," *Proceedings of the third ACM conference on Recommender systems - RecSys 09*, 2009.

[6] "Distance and similarity measures," *SpringerReference*.

[7] M. Grčar, D. Mladenič, B. Fortuna, and M. Grobelnik, "Data sparsity issues in the collaborative filtering framework," in *Advances in Web Mining and Web Usage Analysis* (O. Nasraoui, O. Zaïane, M. Spiliopoulou, B. Mobasher, B. Masand, and P. S. Yu, eds.), (Berlin, Heidelberg), pp. 58–76, Springer Berlin Heidelberg, 2006.

[8] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *Computer*, vol. 42, pp. 30–37, Aug 2009.

[9] D. Powers, “Evaluation: From precision, recall and f-factor to roc, informedness, markedness correlation,” *Mach. Learn. Technol.*, vol. 2, 01 2008.

[10] A. Said and A. Bellogín, “Comparative recommender system evaluation: Benchmarking recommendation frameworks,” in *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys ’14, (New York, NY, USA), pp. 129–136, ACM, 2014.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Neural Information Processing Systems*, vol. 25, 01 2012.

[12] A. Graves, “Generating sequences with recurrent neural networks,” *CoRR*, vol. abs/1308.0850, 2013.

[13] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th International Conference on Machine Learning*, ICML ’08, (New York, NY, USA), pp. 160–167, ACM, 2008.

[14] D. Kim, C. Park, J. Oh, S. Lee, and H. Yu, “Convolutional matrix factorization for document context-aware recommendation,” in *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys ’16, (New York, NY, USA), pp. 233–240, ACM, 2016.

[15] R. Burke *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, p. 331370, 2002.

[16] J. Rongfei, J. Maozhong, and L. Chao, “Using temporal information to improve predictive accuracy of collaborative filtering algorithms,” *2010 12th International Asia-Pacific Web Conference*, 2010.

[17] L. Safoury and A. Salah, “Exploiting user demographic attributes for solving cold-start problem in recommender system,” *Lecture Notes on Software Engineering*, p. 303307, 2013.

[18] "Non-commercial, personalized movie recommendations.."