Generating Negative Samples with a Related Task for Recommendation

by

İpek KIZIL

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

KOÇ
UNIVERSITY

Generating Negative Samples with a Related Task for Recommendation

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

İpek KIZIL

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Assist. Prof. Barış Akgün (Advisor)

_____

Assist. Prof. Mehmet Gönen

_____

Prof. Dr. Şule Gündüz Öğüdücü

Date: _____

to my family

ABSTRACT

Recommender systems benefit both the consumers by helping them navigate a large selection of items and the providers by increasing their profits. Majority of these systems are trained based on consumer preferences such as product ratings/reviews and purchase history. The former is considered as explicit data and the latter is considered as implicit data. Explicit data is easier to work with but is not easy to obtain from each user. Implicit data is immediately available when a user purchases an item and as such majority of the recommender systems use this type of data. The drawback of using implicit data is that it only includes the observed (positive) user-item pairs and the resulting system performance suffer from lack of negative examples. By only looking at the implicit data, it is nearly impossible to interpret why a user did not purchase an item. It may be because the user does not like that item, which implies a genuine negative example, or because the user is simply unaware of it. There are two main approaches to tackle this issue. First one is to consider all the unobserved user-item pairs as missing which leads to a highly sparse data set. The second one is to consider all or a weighted subset of unobserved pairs as negative which causes miss-interpretation by losing user-item pairs that may be positive.

In this work, we propose a novel approach to explicitly deal with this ambiguity, by using a related task, namely click through rate (CTR) prediction. We train a deep neural network based CTR model and use it to generate negative samples for another deep recommendation model. We call our approach Related Task Sampling (RTS). We use real-world datasets from a large tourism company to train both the CTR and the recommendation models in multiple conditions and compare with common baselines. Our main results show that the recommendation models trained by RTS negatives outperform the baselines by more than 10% on average.

ÖZETÇE

Öneri sistemleri, sadece tüketicilere geniş bir ürün yelpazesi sunmakla kalmıyor aynı zamanda tedarikçilerinde karlarını arttırarak büyümelerine fayda sağlıyor. Bu sistemlerin çoğunluğu, ürün değerlendirmeleri (açık veri) ve satın alma geçmişi (örtük veri) gibi tüketici tercihlerine göre eğitilmiştir. Açık verilerle çalışmak daha kolaydır, ancak her kullanıcıdan elde etmek kolay değildir. Çoğu öneri sistemi örtük veri kullanmaktadır çünkü kullanıcı satın alma yaptığı anda tüm bilgileri kaydedilmektedir. Örtük verileri kullanmanın dezavantajı, sadece gözlemlenen (pozitif) kullanıcı-ürün çiftlerini içermesi ve sonuçta ortaya çıkan sistem performansının olumsuz örneklerin eksikliğinden muzdarip olmasıdır. Sadece örtük verilere bakarak, bir kullanıcının bir ürünü neden satın almadığını yorumlamak neredeyse imkansızdır. Bunun nedeni, kullanıcının bu öğeyi beğenmemesi ki bu gerçek bir olumsuz örnek anlamına gelir veya kullanıcının yalnızca ürünün farkında olmaması olabilir. Bu soruna değinen iki ana yaklaşım vardır. Birincisi, tüm gözlemlenmemiş kullanıcı-ürün çiftini bilinmeyen veri olarak düşünmektir ama bu oldukça seyrek bir veri kümesi oluşmasına neden olur. İkinci yaklaşımda gözlemlenemeyen tüm kullanıcı-ürün çiftlerini ya da onların ağırlıklandırılmış ortalamasını olumsuz örnek olarak almaktır. Bu yöntem ilerde olumlu örnek olabilecek kullanıcı-ürün çiftlerini kaybederek yanlış çıkarımlar yapmamıza neden olabilir. Bu çalışmada olumsuz örnek çiftlerini oluşturmak için ikinci bir modelden yararlanan derin öğrenmeye dayalı yeni bir method önerdik. Yaklaşımımızı İlgili Görev örnekleme (RTS) diye adlandırdık. Hem CTR hem de öneri modellerini çoklu koşullarda eğitmek ve temel yaklaşımlar karşılaştırmak için büyük bir turizm şirketinden gerçek dünya veri setlerini kullandık. Başlıca sonuçlarımız, RTS negatifleri tarafından eğitilen öneri modellerinin temel yaklaşımlarına oranla ortalama olarak $\%10'dan$ daha fazla performans sağladığını göstermektedir.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude and profound respect to my advisor Assist. Prof. Dr. Barış Akgün for his continuous support of M.Sc., study and research, for his patience and enthusiasm in his own way. It has been an honour to be his first student. I am thankfull to my committee members, Asist. Prof. Mehmet Gönen and Prof. Dr. Şule Gündüz Öğüdücü for their support and participation politeness.

M.Sc. is a long journey that I will never forget in my life. Listing people name seems to me harder than M.Sc. Instead, I will follow one of my favorite authors acknowledgment solution. There are lots of people in my mind and I always find making a list of people that I want to thank as a difficult task. In this part, I again try to write all the names that are in my memory and the list is longer than I imagine. As I think, names come to my mind and I delete all of them. Instead of doing a list of people; I want to thank all of the people that touch my life either positive or negative. My family, my friends, my teacher, people that show me the way in very dark of my life, my students, my advisees, my advisers, people that love me, people that I hate, people that I dream about, people that I broke their dream, people that I touch their heart, people that touch my heart, people that I upset, people that make me upset. Without these people, I could not be me. Apart from these, I present my gratitude to the Creator who gave me the strength and intelligence that brings me to this level and make me write these lines.

TABLE OF CONTENTS

List of Tables **ix**

List of Figures **x**

Chapter 1: Introduction **1**

1.1 Thesis Statement 3

1.2 Summary of Contributions 3

1.3 Outline 3

Chapter 2: Background **5**

2.1 Recommender Systems Overview 5

2.1.1 Collaborative Filtering 5

2.1.2 Content-Based Recommender Systems 6

2.1.3 Knowledge Based Recommender Systems 6

2.1.4 Demographic Recommendation Systems 7

2.1.5 Hybrid Recommender Systems 7

2.2 Recommender Systems,Challenges 8

2.2.1 Cold Start Problem 8

2.2.2 Lack of Diversity 8

2.2.3 Data-Sparsity 9

2.3 Sampling Methods 10

2.4 SVM and Matrix Factorization 11

2.4.1 Matrix Factorization 11

2.4.2 Support Vector Machine (SVM) 13

Chapter 3: Data Wrangling **17**

3.1 Click Through Rate (CTR) Dataset 17

3.1.1 CTR Dataset Preprocessing 18

3.2 Recommendation Dataset 18

3.2.1 Recommendation Data Wrangling 19

Chapter 4: Generating Negative Samples with a Related Task for Recommendation **22**

4.1 Click Through Rate Prediction (CTR) 22

4.2 Generating Negative Samples with a Related Task for Recommender Systems 23

4.3 Network Architecture and Optimization 24

4.4 Curriculum Learning 27

Chapter 5: Experiments and Results **28**

5.1 Experimental Settings & Evaluation Protocols 28

5.2 Baselines 30

5.3 Results 31

5.3.1 Baseline Comparison 31

Chapter 6: Conclusion and Future Work **35**

Bibliography **37**

LIST OF TABLES

2.1 Hybrid Recommendation Systems Combination Methods 7
2.2 A sample dataset user-hotel purchase 12

3.1 CTR dataset fields. 18
3.2 Recommendation dataset fields. 21

5.1 Comparison of baseline method with Related Task Sampling 32
5.2 Matrix Factorization (MF) Results 33
5.3 HR@N Comparison of baseline method with Related Task Sampling . 33

LIST OF FIGURES

4.1 Stepwise explanation of our method. Step 1: Optimize CTR network with CTR datasets. Step 2: Feed Recommendation data Missing Entries to CTR network to collect likelihood of being negative. Step 3: Use positive data from Recommendation dataset and selected missing entries by CTR Network to train Recommendation Network. 25
4.2 General Network Architecture 26

5.1 Comparison of ROC curves 34
5.2 2-Class SVM Random Sampling 34
5.3 2-Class SVM Related Task Sampling 34

Chapter 1

INTRODUCTION

Nowadays, recommender systems are one of the most important tools of many areas such as marketing, social networking, entertainment, food and tourism industries. From book suggestions to read, songs to try, movies to watch, providing choices on items to buy, restaurants to go, or hotels to stay, recommender systems make it easier for people to make choices. [Iyengar and Lepper, 2000] showed in a real-life experiment that a stand with 24 different kinds of jam only converted 3 customers to a sale, whereas a stand with 6 different kinds of jam converted 30 customers to a sale. They show the effect of guidance over the number of choices available in decision making. As such, we argue that recommender systems help both consumers and providers.

Recommender systems rely on user feedback to make recommendations. There are two types of user feedback which are collected by these systems. The most convenient and high-quality user feedback is external user feedback which includes the user's explicit input regarding her interest in the specific item. For example, Netflix collects movie ratings where high ratings indicate like and low ratings indicate dislike [Bennett et al., 2007]. However, explicit feedback is not always available in many systems since collection of it is laborious. On the other hand, implicit feedback, which entails customer behavior such as purchase history, browsing history, or click-stream data [Kelly and Teevan, 2003], is more abundant and easier to obtain [Y. Hu and Volinsky, 2008].

Although the advantages are clear, a drawback of implicit feedback is that there is no information about what the user did not like, because the unobserved user-item pairs can be interpreted in many different ways [Pan et al., 2008]. For example, the

reason why a user did not purchase an item could be that she was simply not aware of it. In this case, it is hard to interpret a user's preferences by only looking at the observed events that are positive examples of what the user likes.

There are different strategies to handle such ambiguity. One approach is making predictions based on only positive-data by treating all the others as missing [Verstrepen et al., 2017] which is also called "one class scenario". This is an intuitively suboptimal strategy since it attempts to learn only from a very small set of positive examples relative to all the available user-item pairs. The other extreme approach is treating all the unknown user-item pairs as negative examples [Yu and Lin, 2016]. This too seems suboptimal in that a user-item pair that may turn positive in the future would be marked as a negative example. The issues stated above lead to another approach which considers missing data as potential negatives [Pan and Scholz, 2009, Weston et al., 2013, Sindhwani et al., 2010, Li et al., 2010, Paquet and Koenigstein, 2013]. Since the more the users consumed the item, the more likely she enjoyed it. In order to satisfy user needs and achieve personalized recommender system, negative samples should be incorporated into modeling and/or testing for recommendation tasks. Because utilizing only positive instances for modeling the preferences can cause miss-interpretation.

In this thesis, we propose a novel method where a related task namely, Click Through Rate Prediction (CTR) is used to generate missing negative examples to tackle the aforementioned challenges. We randomly generate previously unseen user-item pairs and take the ones with low-likelihood as negative samples for our recommendation task. We then develop and train a deep learning based binary classifier with both existing positive samples and generated negative samples. We show that our approach outperforms the state-of-the-art for a highly sparse problem and argue that this will lead to better recommender systems.

1.1 Thesis Statement

A deep learning based negative sampling method that turns positive only data problem into a binary classification problem for recommender systems by using a related task[1] increases recommender system performance compared to prior research.

1.2 Summary of Contributions

The following contributions are made in this thesis:

- We developed a deep learning based method to generate missing negative samples for an implicit feedback hotel recommender system by using a related task, namely click through rate (CTR) prediction.
- We developed a deep neural network based binary classifier to predict likes and dislikes of a user and train this model with implicit (positive) and generated (negative) data.
- Use two large real-world datasets, one explicit dataset for CTR and the other implicit dataset for customer behavior, to perform extensive experiments to demonstrate the performance of both our negative sampling method and our user-like classifier. We show that our proposed methods significantly outperform the state-of-the-art.

1.3 Outline

The remainder of this document is organized as follows

- Chapter 2 **Background** provides the background knowledge for recommender systems and respective literature survey of prior research. It presents challenges and proposed solutions. It also includes, brief information about Support Vector Machines and Matrix factorization and how we applied them in this study.

[1]Related task refers to Click Through Rate Prediction that uses explicit click stream data in this thesis.

- Chapter 3 **Data Wrangling** presents recommendation and CTR dataset fields and discusses the methodologies that are used generate these data fields.
- Chapter 4 **Generating Negative Samples with a Related Task for Recommendation** details the developed deep learning based recommender system and the proposed method to generate missing negative examples for the recommendation system.
- Chapter 5 **Experiments** explains the experiments, results and analysis of proposed approach.
- Chapter 6 **Conclusion and Future Work** discusses the conclusions and contributions of this thesis and presents future directions.

Chapter 2

BACKGROUND

2.1 Recommender Systems Overview

Traditionally, we can categorize recommender systems into three types based on their input as follows: collaborative filtering based recommender system that is the system uses correlations of user-item interactions, content-based recommender systems either uses item features to match with users or item-item correlations to make recommendations. Both of the systems suffer from their own drawbacks. The hybrid recommender systems deal with the limitations of collaborative and content based systems by combining them in different ways [Chu and Tsai, 2017]. Lately, with the increasing attention from the academia and the industry for the recommender systems added new extensions to the existing approaches such as: demographic filtering based recommender systems where preliminary information of users and groups are used to build a common profile that matches with the existing items, knowledge-based recommender systems which incorporate the knowledge of user's preferences and need to make the recommendation. Also another categorization of the recommender systems can be made upon the output type such as: ranking based where the items are ranked according to user preferences, rating based which use the ratings given by user to the item, and classification based which aims to classify items into categories [Shuai Zhang, 2017].

2.1.1 Collaborative Filtering

Collaborative filtering is one of the most famous approach used in personalized recommender systems. There are three forms of collaborative filtering: user-based, item-based, and matrix factorization based. In the first user-based approach, the idea is

that people influence each other either implicitly or explicitly. Thus, we could infer user preferences by looking at other similar users [Y. Hu and Volinsky, 2008, He et al., 2017]. The second approach, called item-based collaborative filtering, which can be thought as the permuted version of user-based approach where similar items are recommended to the users who already bought or liked them [Deshpande and Karypis, 2004]. The last approach, matrix factorization, relies on the idea that we need user-item rating matrix to model user preferences. By decomposing this matrix, it creates more compact representation of the user preferences to predict how likely a user may choose an item which she has not rated before based on other users' preferences [Aghdam et al., 2017].

2.1.2 Content-Based Recommender Systems

Content-based recommender systems use content information of the items to model user preferences and make recommendations based on the predicted preferences [Musto et al., 2016]. The content information used to describe the items can diverge such as texts, images [Rawat and Kankanhalli, 2016] and videos [Chen et al., 2017a]. Each user profile is generated independent of other users, unlike the collaborative filtering based approaches. For example, even if two users share the same taste from the same category, if one of the users did not purchase anything from that category, system would not suggest any item from the common category. One of the drawbacks of the content based approach is it requires a significant amount of feature engineering to collect item content features. On the other hand, the cold start problem for a new item is solved by using the content information of the item.

2.1.3 Knowledge Based Recommender Systems

In contrast to the other systems, knowledge based recommender systems does not rely on the user-item interactions as a main source. They use a prior knowledge of how a particular item meets a particular user's needs [Aggarwal, 2016]. Here, the knowledge might be about the users, items or user-item pairs. [Burke, 2000] uses the knowledge

about the cuisines, foods, and restaurants to make restaurant recommendation to its users. The main drawback of the knowledge based systems is that they require domain expertise.

2.1.4 Demographic Recommendation Systems

Demographic recommender systems use demographic information of the users such as age, country, city etc. to categorize them and make recommendations based on predefined user classes [Zhao et al., 2015, Katarya and Verma, 2015, Al-Shamri, 2016]. As a result, demographic recommendation systems form user-user correlations, but use different data. Demographic recommender systems have the advantage that they do not need user ratings needed by the collaborative recommender systems or content information needed by the content based recommender systems. The main drawback of the demographic recommender is that it is hard to collect users' demographic information.

2.1.5 Hybrid Recommender Systems

All of the aforementioned recommender systems have some drawbacks. Hybrid recommender systems focus on the idea of combining multiple recommender systems to overcome these drawbacks to make recommendations. There are several ways to create hybrid recommender systems [Burke, 2002] as shown in Table 2.1.

Table 2.1: Hybrid Recommendation Systems Combination Methods

Combination Method	Description
Weighted	The scores of several recommendation techniques are combined together to produce a single recommendation.
Switching	Depending on the case, the system switches between the recommendation techniques
Mixed	All the recommendations are presented at the same time from all the recommendation systems
Feature combination	Features of different recommendation data sources are used in a single recommendation algorithm.
Cascade	One recommendation system refines the recommendations given by another.
Feature augmentation	Output of one recommendation system is used as an input for the other recommendation system.
Meta-level	The model learned by one recommendation system is used as input for the other recommendation system.

2.2 Recommender Systems,Challenges

2.2.1 Cold Start Problem

The cold-start problem refers to the beginning stage of launching a recommender system within which the system might not be able to provide useful recommendations, mainly due to the lack of enough data [Adomavicius and Tuzhilin, 2005]. If a user is new to the system mainly we lack the taste of the user. The same is applicable for the items. One of the approaches for managing the cold-start problem is using meta-data or socio-demographic data to infer the profiles [Katarya and Verma, 2015]. Another way would be employing external data, e.g. other usages of the items that are better known, to make the recommendations [Kim et al., 2017]. The external data can be used until the system gathers enough data about the objects.

In this study, since the data is really sparse we used the content information of the hotel and the user information to overcome cold-start problem. The best solution is usin the hybrid system which we applied to provide recommendations.

2.2.2 Lack of Diversity

Collaborative filtering is the most common algorithm that used in the recommender systems. We could define collaborative filtering in two ways. User-based collaborative filtering that rely on user-user interactions that similar users share the similar taste and the item-based that rely on item-item similarities to make recommendations where always similar items are recommended to the user. This situation causes lack the diversity. To overcome this problem, the best way to build a recommender system is using the user-item interactions instead of using user-user or item-item similarities. Deep learning is best approach to model user-item interactions which reduces feature engineering workload.

2.2.3 Data-Sparsity

Sparsity is the problem of lack of information. Suppose we have a huge amount of users and items but user have rated only few items. If a user has evaluated only few items then its difficult to determine the user preferences. To tackle this issue hybrid approach will be the best solution.

2.3 Sampling Methods

The main challenge of all the recommender systems is modeling the user preferences. The user preferences are modeled based on user feedback collected by the system. So, the type of gathered user feedback plays a vital role in development and evaluation of the recommender systems. For example, Netflix collects [Bennett et al., 2007] ratings given by user for its content while amazon collects what user purchased/clicked [Linden et al., 2003]. The latter example is referred to as implicit feedback which only includes what a user purchases. On the other hand, in the former example, which is referred to as explicit feedback is directly provided by the user. Explicit feedback tells whether the user likes or dislikes the item. Since the user's likes positive preferences, and dislikes, negative preferences are observed directly, most of the successful recommender systems use explicit feedback [Musto et al., 2016]. However, it is hard to collect explicit feedback in many systems, especially in the domain of small e-commerce, because of the system limitations and the necessity of the user's involvement. Many people are not willing to spend so much time providing explicit feedback. The challenges in collecting the explicit feedback dictates the need for the use of implicit feedback in recommender systems [Yu et al., 2017]. Implicit feedback is common to many systems and easily available. One of the drawback of the implicit feedback is that only positive user preferences are obtained. It does not have information why a user did not purchase an item; whether she did not like it or simply she was not aware of it. As a result, implicit feedback lacks negative examples. Thus, ambiguity arises while developing recommender systems that use implicit feedback.

There are three major ways of dealing with missing negative examples. One way is to only consider positive samples for training which turns the problem into one-class classification [Nati and Jaakkola, 2003]. The drawback of this method is that it is possible to learn a trivial solution which outputs positive no matter what the input is. This type of methods can achieve high recall, but low precision.

Another scenario is to consider all missing samples as negative. This sampling strategy further branches into two major categories sub-sampling a negative set of

instances with similar size to the positive instances or treating all of the missing instances as negative. The latter approach is costly due to the dataset size increase. As a result, [Yu and Lin, 2016] come up with an efficient way to use all samples instead of sub-sampling. However considering missing data as negative has its own drawbacks since it is not really indicative of users' negative preferences about the missing items, because of lack of awareness about the item.

The issues stated above leads to another approach which considers missing data as potential negatives [Pan and Scholz, 2009, Weston et al., 2013, Sindhwani et al., 2010, Li et al., 2010, Paquet and Koenigstein, 2013]. [Pan et al., 2008] uses several weight schemes which considers either using item or user information from the datasets. Also, they weighted missing entries less since their true labels are not known. [Y. Hu and Volinsky, 2008] uses a similar method which weights matrix factorization with confidence. Confidences are inferred based on how many times the user consumed a specific item. The more the user consumed an item, the more likely she enjoyed it. [Zhang et al., 2013] also applies the weighting approach but different than the other approaches. Their approach uses the position of the missing data in the sampled list to rank the weights.

2.4 SVM and Matrix Factorization

In this section, we introduce the matrix factorization method that relies on positive only data and one of the best classification methods, support vector machine (SVM). Later we will discus, how we applied these methods and present their performance in Chapter 5.

2.4.1 Matrix Factorization

The idea behind matrix factorization is to figure out how the users and the items are related to each other by decomposing the user-item ranking matrix R of size $M \times N$,where M is the number of users, and N is the number of items into two low-rank matrices. First one is a $M \times K$ matrix which represents the latent user feature

Table 2.2: A sample dataset user-hotel purchase

	Hotel-1	**Hotel-2**	**Hotel-3**
User-1	2	-	-
User-2	-	-	1
User-3	-	1	-

vectors for each user and the second one is a $K \times N$ matrix represents the latent item feature vectors for each item [Koren et al., 2009]. Multiplying these two dense matrices that includes a number of latent features K for each of our items and users approximates the original rating matrix Equation 2.1 :

$$R_{ij} \approx M_i N_j^T \tag{2.1}$$

By using the approximated user-item matrix, this method aims to predict unobserved item ratings of users.

We can define the rating matrix rows as corresponding to the users and its columns corresponding to the items. The entries of the rating matrix is the ratings given by the users to the items. In our case, since we lack explicit user ratings for each item, entries indicate the number of times an item was purchased by a specific user. The blank entries indicate that there was no interaction between user and item. In our dataset items correspond to hotels. Table 2.2 shows a sample dataset of users, hotels, and hotel purchases. It is worth to mention that, our real matrix is very sparse since most users only interact with a few hotels.

As discussed earlier, the objective of the matrix factorization is filling the missing elements of the sparse data matrix. The biggest challenge in this case is filling in a column or a row of the matrix that has very few user-item interactions. To deal with this challenge, one way is incorporating user/item features, often referred to as side information. For example, in our hotel recommendation system, it could be the city where the hotel is located and the gender of the user. Side information is crucial

for predicting sparsely observed user-item relations. In this study, direct application of matrix factorization for hotel recommendation failed since the %99.9 of the data is sparse. We also used user and hotel side information to improve the performance of the recommender system. We evaluated the performance of matrix factorization based on different scenarios as explained in Chapter 5.

2.4.2 Support Vector Machine (SVM)

Support Vectors Machines (SVM) is one of the most commonly used classifiers which tries to find the best hyperplane to separate different classes by maximizing the distance between sample points close to the hyperplane. It commonly uses a technique, called the kernel trick, to transform the data and based on these transformations, it finds an optimal boundary between the classes.

SVM implements a classification approach[Smola and Schölkopf, 2004]:

$$f_{\text{nonlin}}(\mathbf{x}) = \langle x, w \rangle_2 + b = \sum_{k=1}^{n} w_k * x_k + b \qquad (x \in R^n) \tag{2.2}$$

with unknown, but fixed, $\mathbf{w} \in R^n$ and $b \in R$. These SVM classification parameters (or level 1 parameters) are determined during SVM training on given data. Once these parameters are adjusted, binary classification for any $x \in R^n$ is achieved via a hypothesis function

$$h(x) = sgn(f(x)) \tag{2.3}$$

where sgn() is the modified signum function defined as

$$sgn(a) = \begin{cases} 1, & \text{if } a \geq 0. \\ -1, & \text{else.} \end{cases} \tag{2.4}$$

If the dataset under consideration is not linearly separable, a nonlinear function $\phi : r^n \to D$ is used to map the data to a space D of possibly very high dimension $d \in N$. This leads to

$$f(\mathbf{x}) = \langle \phi(x), w \rangle_D + b = \sum_{k=1}^{n}) w_k * \phi(x_k) + b \qquad (x \in R^n) \tag{2.5}$$

where Φx denotes the d-dimensional vector resulting from the nonlinear transformation of $\mathbf{x}$, and $\mathbf{w} \in \mathbf{D}$,$\mathbf{b} \in \mathbf{R}$ are the classification parameters. In a so called soft margin model [Cristianini and Shawe-Taylor, 2000] that allows for training errors ξ, the classification parameters can be derived from the solution of the constrained convex optimization problem.

$$\left.\begin{aligned} \min_{w \in D, b \in R, \xi \in R^l} & \frac{1}{2}\|\mathbf{w}\|_D^2 + C \sum_{i=1}^{l} \xi^q, \\ s.t. \qquad & y_i \cdot f_{nonlin}(\mathbf{x}^i) \geq 1 - \xi_i, i = 1, ..., l, \\ & \xi_i \geq 0, i = 1, ..., l \end{aligned}\right\} \tag{2.6}$$

Here, $\{(\mathbf{x}^i, y_i) \in R^n \times \{-1, 1\}, i = 1, ..., l\}$ is a set of given input - output data pairs (or training points).$q \in N_+$ is used to define the influence of the so-called $\xi_i (i = 1, ..., l)$ slack variables. SVM soft margin methods are divided into and models[Cristianini and Shawe-Taylor, 2000].

A special feature of support vector learning is the implicit nature of the mapping ϕ.

$$\left.\begin{aligned} & \min_{\alpha \in R^l} g(\alpha) := \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j \langle \phi(x^i), \phi(x^j) \rangle_D - \sum_{i=1}^{l} \alpha_i, \\ s.t. \qquad & \langle \mathbf{y}, \alpha \rangle_2 = 0 \\ & 0 \leq \alpha_i \leq C_i, i = 1, ..., l \end{aligned}\right\} \tag{2.7}$$

For SVM learning typically a single upper bound $C_i \equiv C \in R_i$ is used for all Lagrange multipliers $\alpha_i (i = 1, ..., l)$. Only the dot products between data points in the high-dimensional space D occur; the points $\phi(x^i)$ themselves are not required. Thus substituting the dot products with the values $K(x^i, x^j)$ of a nonlinear kernel

$K : R^n \times R^n \rightarrow R$, relieves the developer from constructing an explicit nonlinear mapping for the input data. Often the Gaussian kernel (radial basis function kernel)

$$K^G(x^i, x^j) = \exp(-\frac{\sum_{k=1}^{n}(x_k^i - x_k^j)}{2\sigma^2}) \tag{2.8}$$

is used in the context of SVM training.

Letting $Q_ij = y_iK(x^i - x^j)y_j$,$1 \leq i, j \leq l$, the dual objective function can be written as

$$g(\alpha) := \frac{1}{2}\alpha^T Q\alpha - \sum_{i=1}^{l}\alpha_i \tag{2.9}$$

with a positive semi - definite Matrix Q. Since we deal with convex problems the existence of unique global solutions are guaranteed. Furthermore the optimal function values are equal. It turns out that the resulting dual solution α^* , the vector of Lagrange multipliers, can be used to define a dual classifier [Cristianini and Shawe-Taylor, 2000]

$$\begin{aligned} f_{\text{nonlin}}(\mathbf{x})^* &= \sum_{k=1}^{d} w_k^*\phi_k(\mathbf{x}) + b^* = \sum_{k=1}^{d}\left(\sum_{i=1}^{l} y_i\alpha_i^*\phi_k(\mathbf{x}^i)\phi_k(\mathbf{x}) + b^*\right) \\ &= \sum_{i=1}^{l} y_i\alpha_i^*\langle\phi(x^i), \phi(x^j)\rangle_D + b^* = \sum_{i=1}^{l} y_i\alpha_i^* K(x^i, x) + b^* \end{aligned}$$

The threshold b^* can be determined using the Karush KuhnTucker(KKT) conditions [Cristianini and Shawe-Taylor, 2000]. It is well known that only a small number of the training points is located within the margin or on the wrong side of the separating hyperplane. Only these points, called support vectors, have positive Lagrange multipliers and contribute to the final classifier

$$f_{final}(\mathbf{x}) = \sum_{i\in(1,\ldots,l):0<\alpha_i} y_i\alpha_i^* K(\mathbf{x}^i, \mathbf{x}) + b^* \tag{2.10}$$

Support Vector Machines have a range of hyperparameters which strongly affect the predictive performance of the model. One of them, $C \in R^+$, is an important

quantity for the trade-off between margin maximization and error toleration. Usually large values for C lead to few training errors (and a narrow margin), whereas small values generate a larger margin, at the cost of more errors and more training points situated inside the margin. This knowledge does not really help in choosing a suitable value for the parameter since the number of training errors cannot be interpreted as an estimate for the true risk. For good generalization properties emphasis has to lie on a not-too-narrow margin, too. The other set of hyper - parameters are the kernel parameters of Equation 2.8.

The choice of appropriate learning parameters is a crucial step in obtaining well-tuned support vector machines. Usually, the settings of these parameters are based on a grid search [Chih-Wei et al., 2003]. That is, for each parameter a finite number of possible values is prescribed, and then all possible combinations of these values are considered to find the one combination that yields the best result. Therefore, the complexity of grid search grows exponentially with the number of parameters.

In this study, we used SVM to show that our proposed approach is not specific to deep learning based classifiers.

Chapter 3

DATA WRANGLING

The performance of the proposed methods in this thesis is evaluated on two different real-world datasets which are collected from a large travel company in Turkey. The first dataset is called called the Click Through Rate (CTR) dataset which is used for training a CTR network to tailor the negative instances of the recommendation system. The second and main dataset is called the recommendation dataset that is used for training the recommender system. The details about the data and data wrangling steps are discussed in this section.

3.1 Click Through Rate (CTR) Dataset

In this thesis, CTR dataset is used as an explicit data source to learn missing negative examples in recommendation dataset since it consists of the positive examples of user-hotel pairs. In general, the click through rate history consists of only the positive examples, teh user clicks. It does not consist of information about what users did not click; negative examples. For our CTR dataset, we were able to collect which hotel is clicked by the user or not by utilizing another vendor company. In the data collection scenario, the campaigns that include hotel links generated by the travel company is sent via email to the customers. The logs of user's click through history is captured by the vendor company.

This dataset consists of features of user-hotel pairs and the click label. The columns explained in the recommendation data are applied to the CTR data as well. The additional features of CTR dataset is discussed in the Table 3.1. Hotel entities are shown in Table 3.2.

Table 3.1: CTR dataset fields.

Feature Name	Feature Description	Feature Data-Type
EMAIL	Customer email address	string
OS_NAME	Operating system name	string
OS_VERSION	Operating System version	string
BROWSER	Browser type	string
BROWSER_VERSION	Version of the browser	string
DEVICE_NAME	Type of the device where customer reach the system	string
READ_OR_CLICK	Either customer click given hotel page or just read	boolean
CAMP_ID	Campaign id given by the vendor system	string
CAMP_NAME	Campaign name given by the travel company	string
CAMP_DELIVERY_DATE	Campaign delivery date	timestamp
CLICK_STATE	1 click, 0 not click	boolean

3.1.1 CTR Dataset Preprocessing

CTR dataset has two distinct sources owned by different companies. The structure of the data is different on both sides. The click data on the vendor company is stored in JSON format. On the other hand, the travel company stores its own data in a relational database. This requires an enormous amount of detailed steps to combine datasets.

3.2 Recommendation Dataset

Recommendation dataset is our main dataset where we use it to develop the hotel recommender system. It has several relevant distributed sources: hotel data storage (CMS Hotel Systems), user data storage (CRM) and user transactions storage.

The CMS Hotel Systems contain several relevant sub-systems that hold hotel information. Division of the hotel information on the sub-systems are decided according to entities of the hotels, and company's policies. All of the tables for the entities have an ID that comes from the service they are stored and a common unique identifier code generated by main systems to unify all of them. Combining all of the hotel

entities often require a large amount of work in proportion to the number of joins are applied on different services, because the only end combiner is the main identifier code. This code is unique for each hotel on the system. We can list the hotel entities as main hotel attributes, rooms, prices, geo-location, hotel reviews, hotel options, hotel services, and so on.

The user transaction storage, which refer to as the reservation database, comprises a history of user actions that relate to booking, buying, cancellation, and inquiry. Each table hold different entities of user interactions with the items denoted by the user's specific action, e.g. a user made a hotel reservation via call center triggers the booking system to create a new reservation whereas the cancellation refers to labeling the reservation with cancellation code. Inquiry provides different aspects of user's information, e.g. which hotel type is user interested, what is the price range for the user and how the inquiry ended up. Buying refers to the payment details. Reservation database is similar to the CMS Hotels Systems. Reservation has entities such as action, booking history, close code and so on. All of these entities has specific attributes relate to reservation knowledge. All the tables are combined with a unique reservation id. Reservation database joined with other databases by the other's unique key identifiers, e.g. reservation and CMS Hotel Systems are combined by CMS Hotel Systems' main code. Reservation system data consists of both categorical and continuous features that were used in this research. Table 3.2 provides schema of the features that was used in this research.

3.2.1 Recommendation Data Wrangling

Recommendation data has three distinct sources: physical travel agencies, company website and call center. All the data from the given sources are user generated and stored on distributed database systems. Therefore, many data wrangling techniques are applied to the dataset. First, we imported the whole data from multiple sources and collected on HIVE[3] and explored the meaningful common column descriptions to

[3] `https://hive.apache.org`

label and join data with the other systems. Since the written alphabet was Turkish, we had to modify the data entries with special characters into universal format.

Table 3.2: Recommendation dataset fields.

Feature Name	Feature Description	Feature Data-Type
INTERNAL_CONTACT ID	Anonymized customer id	string
VENDOR_CODE	Anonymized hotel main code	string
CITY	City customer is located	string
COUNTRY	Country of residence customer located	string
AGE	Age of the customer generated from birth-date	int
PRODUCT_NAME	Room type what user reserved	string
REGION_NAME	Location of the hotel booked	string
GENDER_CODE	Gender of the customer: female,male and unknown(not specified)	int
CUSTOMER_TYPE	defines the customer type for the company	string
BALAYI OTELI	1 when a customer choose honeymoon hotels, 0 otherwise	boolean
HAYVAN DOSTU	1 when a customer choose animal friendly hotels, 0 otherwise	boolean
OZEL KOYLU OTEL	1 when a customer choose hotels with special bay, 0 otherwise	boolean
SPA TESISI	1 when a customer choose hotels has SPA, 0 otherwise	boolean
PLAJI KUM	1 when a customer choose hotels with sand beach, 0 otherwise	boolean
YILBASI ETKINLIGI OLAN TESIS	1 when a customer choose hotels has new year attraction, 0 otherwise	boolean
DENIZI TASLI OTEL	1 when a customer choose hotels with stony sea, 0 otherwise	boolean
OZEL SAGLIK TALEBI	1 when a customer choose hotels have concierge doctor, 0 otherwise	boolean
TERMAL OTEL	1 when a customer choose thermal hotels, 0 otherwise	boolean
H. LIMANINA YAKIN OTEL	1 when a customer choose hotels close to the airport, 0 otherwise	boolean
YESIL YILDIZ	1 when a customer choose fresh air field hotels, 0 otherwise	boolean
T. SANDALYEYE UYGUN	1 when a customer choose hotels have wheelchair access, 0 otherwise	boolean
GENC DOSTU OTEL	1 when a customer choose youth friendly hotels, 0 otherwise	int
DENIZI KUM CAKIL	1 when a customer choose hotels have sea with sand and gravel, 0 otherwise	boolean
YESIL ANAHTAR	1 when a customer choose green-friendly hotels, 0 otherwise	boolean
GOLF OTELI	1 when a customer choose hotels have golf range, 0 otherwise	boolean
YAZA OZEL ETKINLIGI OLAN OTEL	1 when a customer choose hotels have summer attractions, 0 otherwise	boolean
AQUAPARK OTELI	1 when a customer choose honeymoon hotels, 0 otherwise	boolean
YETISKIN OTELI	1 when a customer choose adult friendly hotels, 0 otherwise	boolean
DENIZE SIFIR	1 when a customer choose seafront hotels, 0 otherwise	boolean
CEVRE DOSTU	1 when a customer choose nature-friendly hotels, 0 otherwise	boolean
KONGRE OTELI	1 when a customer choose honeymoon hotels, 0 otherwise	boolean
SPA/TERMAL OTEL	1 when a customer choose honeymoon hotels, 0 otherwise	boolean
DENIZI KUM	1 when a customer choose congress hotels, 0 otherwise	boolean
DENIZI SIG OTEL	1 when a customer choose hotels with shallow sea, 0 otherwise	boolean
COCUK DOSTU	1 when a customer choose kid-friendly hotels, 0 otherwise	boolean
DOGA OTELI	1 when a customer choose nature hotels, 0 otherwise	boolean
KAYAK OTELI	1 when a customer choose ski hotels, 0 otherwise	boolean
ILERI YAS DOSTU OTEL	1 when a customer choose elderly friendly hotels, 0 otherwise	boolean
MAVI BAYRAKLI	1 when a customer choose honeymoon hotels, 0 otherwise	boolean
BEBEK DOSTU	1 when a customer choose baby-friendly hotels, 0 otherwise	boolean
DENETIMDE BASARILI	1 when a customer choose hotels successful in supervision, 0 otherwise	boolean
AILE OTELI	1 when a customer choose family hotels, 0 otherwise	boolean
RESORT OTEL	1 when a customer choose resort hotels, 0 otherwise	boolean
IS_GONE	1 when a customer chooses a hotel, 0 otherwise	boolean

Chapter 4

GENERATING NEGATIVE SAMPLES WITH A RELATED TASK FOR RECOMMENDATION

In this chapter, the related task used to generate negative samples called Click Through Rate Prediction(CTR) and the methodology of our research is presented. We discuss the relation between recommender systems and CTR and detail our methodology.

4.1 Click Through Rate Prediction (CTR)

Click Through Rate Prediction(CTR) plays an important role for many web applications such as display advertising, web search and recommender systems since it is a valuable source of users' implicit relevance feedback [Wang et al., 2016]. In CTR, the goal is to predict how likely a user is going to click on a web-page when it is sent to her. The web-pages clicked by the user represents the positive examples of user's interests and the pages not clicked by the user are ambiguous. Because we cannot clearly state why a user did not click the page. We do not know whether she did not like it or simply she was not aware of it. The same problem exists for the purchase data as well.

Traditionally, click stream and purchase data do not consist of users' negative response. But different than traditional approaches, the data click-stream data used in this thesis consists of what a user did not click. The click behavior of the user is followed by an external system through their email. Assumption here is that if a user does not click a hotel that we sent her through email, she is not interested in that hotel. If a user clicks a hotel, this shows her interest for that hotel. Since it is useful to have, both users' positive and negative responses for recommendation, we

propose to learn negative user preferences which is missing in recommender system that rely on purchase history of users by using another related task where negative user responses exist.

4.2 Generating Negative Samples with a Related Task for Recommender Systems

The main task for recommender systems is to provide personalized recommendations that rely on user preferences. To model user preferences in recommender systems we usually have two equally important ratings: positive(what user likes/purchase) and negative(what user dislike/might not purchase). Even though both ratings are important, we are only able to observe positive ratings in many recommender systems, because increasing demand for the quick search and the fact that people are reluctant to provide explicit feedback due to their limited time makes it hard to collect the negative examples. Thus, in many cases recommender systems needs to infer user preferences from the implicit user behavior such as browsing history, click through history or the purchase history. The clicks and especially the purchase are good indicators of what user liked (i.e., positive training examples), but the items a user did not like (negative training examples) are not directly observed. Previous approaches either randomly pick negative training samples from unseen items [Nati and Jaakkola, 2003] or incorporate some heuristics [Pan et al., 2008, Pan and Scholz, 2009, Hu et al., 2008, Sindhwani et al., 2010] into the learning model, leading to a biased solution and a prolonged training period. The latter one uses weighting for missing entries to decide how likely they are true negative examples.

Here, we propose a a deep learning based hotel recommender system which exploits the implicit purchase history. Due the fact that the recommender systems with explicit feedback are the most successful approaches and they give better recommendation, we seek negative examples for out recommender system. We propose a novel method based on the assumption that negative examples for the recommender system can be estimated from a related task with explicit feedback. The related task

to the recommender system mentioned here is the click through rate prediction of the users. Usually the click through rate history does not consist the not click behavior of the user, but here the way how the click through data is collected provide information about what user did not click. Since people go on vacation rarely, we can assume that which hotel they clicked or not indicates whether they are going to purchase that hotel or not. Based on the fact that explicit feedback consists of the user's negative and positive preferences, we developed a 3 step approach to tailor our negatives instances required by the recommender system using an explicit parallel task. Our 3 step approach is depicted in Figure 4.1. The first step, learns a CTR network with explicit CTR dataset. In the second step, learned CTR network is used to extract likelihoods of missing entries being negative from recommendation dataset. The third step uses the positive data from recommendation dataset and its missing entries selected as negative by CTR network to train a recommender system. We call this approach Related Task Sampling (RTS).

The logits of CTR network is interpreted as degree of or likelihood of being negative examples. Positive logit values indicates the input tends to be positive, while negative values indicates, it tends to be negative. Positive logit values with higher values are more likely to be positive, while negative logits with lower values are more likely to be negative. In this work we chose logits smaller than 0 as negative samples without using their absolute value as weighting strategy.

4.3 Network Architecture and Optimization

To predict user preferences we have developed a Multi Layer Perceptron (MLP) which models user-hotel interactions as an classification problem (Figure 4.2). Overall, this network takes in hotel-user input pair and outputs a class probability which represents how likely a user select a specific hotel.

Each layer corresponds to an affine transformation (linear matrix multiplication) followed by a non-linearity. Output of each layer feeds into input of the next layer. All layers receive single feature vector as input except the first one which takes two;

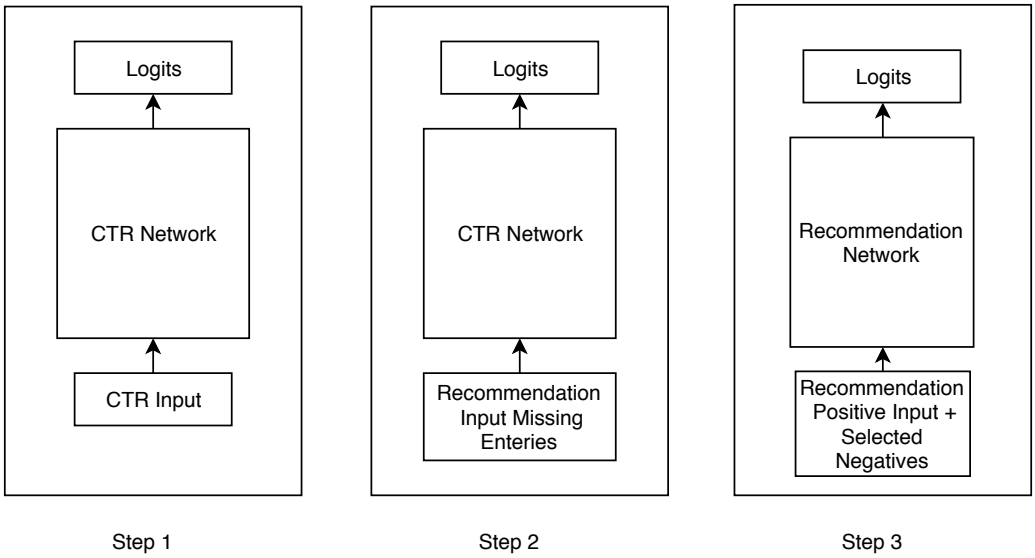

Figure 4.1: Stepwise explanation of our method. Step 1: Optimize CTR network with CTR datasets. Step 2: Feed Recommendation data Missing Entries to CTR network to collect likelihood of being negative. Step 3: Use positive data from Recommendation dataset and selected missing entries by CTR Network to train Recommendation Network.

user's feature vector $\boldsymbol{U_i}$ and hotel's feature vector $\boldsymbol{H_i}$. The first layer applies embedding, will be explained shortly, on each input vector separately then the outputs are concatenates into a single vector. The concatenated vector is feed into 4 consecutive layers. Number of hidden units in each layer is selected as 64 by hyper-parameter tuning from a set $\{64, 128, 256\}$ on validation set. ReLU [Nair and Hinton, 2010] is used as non-linearity in hidden layers and for the last layer sigmoid is used to output class probability. Weights in each layer are initialized with Glorot initializer [Glorot and Bengio, 2010] while the biases are initialized with zeros.

Both user and hotel input vectors are highly sparse and high dimensional. We have used embedding layers to map them into low dimensional dense vectors. The embedding layers are basically matrix multiplication without non-linearities. This

method is similar to word embeddings of [Guo and Berkhahn, 2016], however we learn the parameters end-to-end together with other network parameters instead of pre-training on another task. As embedding size, output dimension size, we have used 10 after a hyper-parameter search from a set {10, 20, 50, 100} on validation set.

We use user specific information as an input and the content information of the hotel. Thus generic information of the user and the hotel is used to adjust the cold start problem in recommender systems.

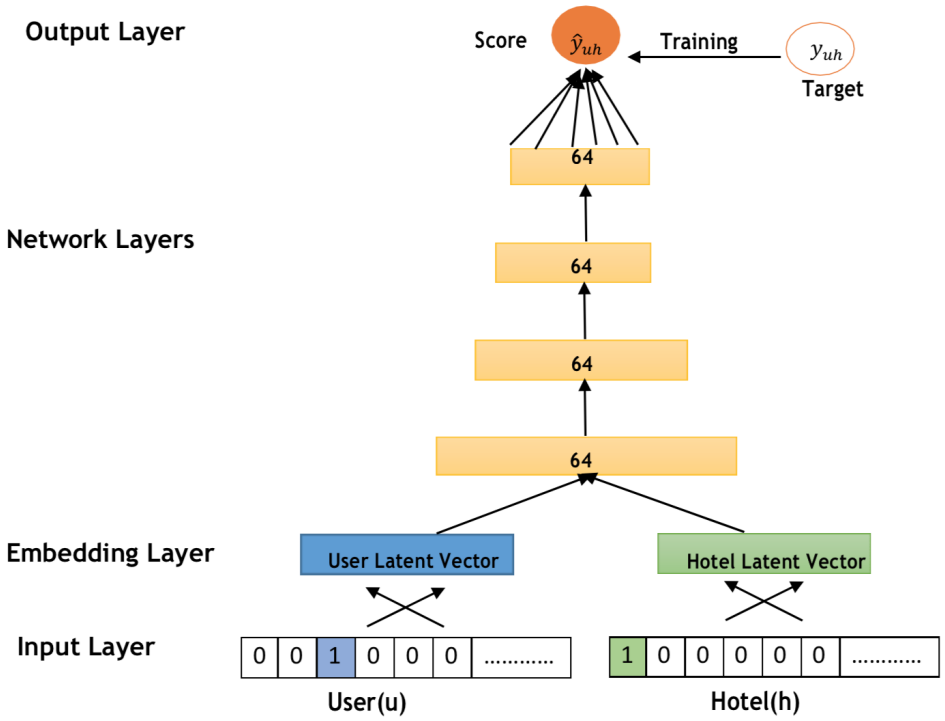

Figure 4.2: General Network Architecture

The objective of the model is cross entropy and optimizer is ADAM with learning rate of **0.0001** and default $\boldsymbol{\beta}$ values [Kingma and Ba, 2014]. The network is trained for 32k iteration for each experiment.

As our data both CTR and recommendation datasets naturally comes with imbalanced classes we have used sub-sampling approach to balance them. For each iteration we have selected 50 positives and 50 negatives which constitutes batch size

of 100.

4.4 Curriculum Learning

As an extension to this method we have used curriculum learning [Yoshua et al., 2009] to further exploit logits of the CTR network. Curriculum learning starts from easy examples to train the model and gradually introduces harder examples. Basic intuition behind this idea is, the network can be optimized easier with simple examples first, then this learned network can be a initialization for harder examples. As a result, we expect the network optimization to get easier. [Yoshua et al., 2009] has used heuristics to choose what is easy and what is hard. In our case, we have considered the CTR network to find out the easiness level of an example and used them in curriculum learning. This is a novel combination of parallel task and curriculum learning.

Chapter 5

EXPERIMENTS AND RESULTS

We conducted our experiments on the two real-world datasets from a large tourism company of Turkey as detailed in Chapter 3. We evaluated our model performance against multiple baselines. In addition, we have used our method together with SVM and matrix factorization to show that the performance of our proposed method is not limited to neural networks.

5.1 Experimental Settings & Evaluation Protocols

We experimented our method by using two real-world datasets from a tourism company of Turkey.

1. **Recommendation dataset**. This implicit feedback dataset consists of users' hotel purchase history. The original data was very large and consists of purchase history of twelve types of hotels. In our experiments, we used a subset that contains one million purchase interacts of one type of hotel: holiday villages, where **80%** of the users have only 1 purchase. It only contains information about which the users purchased.

2. **Click through rate (CTR) dataset**. Since, it is an explicit feedback data, we have chosen it to approximate our negative samples. Later, we used negative samples to evaluate the performance of the our sampling method against other approaches. The label of each entry marked as 0 or 1 indicating whether the user has clicked the hotel or not.

Training/test split is 80/20 with random selection. All experiments were run for 10 times and confidence scores are provided. To evaluate the performance of our

sampling method we adopted several error and performance metrics such as; precision, recall, the area under the ROC curve (AUC). Recall is defined as:

$$\text{True Positives}/(\text{True Positives} + \text{True Negatives}) \tag{5.1}$$

and Precision is defined as:

$$\text{True Positives}/(\text{True Positives} + \text{False Positives}) \tag{5.2}$$

False positives are cases the model incorrectly labels a negative example as positive, i.e. ground-truth of the hotel-user pair is "has gone", but the model predicted it as "has not gone". While recall expresses the ability to find all relevant instances from the dataset, precision expresses the portion of relevant instances among retrieved instances. Our third metric is the area under an ROC curve (AUC) which falls between 0 and 1 with a higher number indicating better classification performance. AUC as the name indicates, quantifies the a model's ROC curve by calculating the total Area Under the Curve (AUC). A ROC curve shows how the recall vs precision relationship changes as we vary the threshold for identifying a positive in our model. The threshold represents the value above which a data point is considered in the positive class. A ROC curve plots the true positive rate on the y-axis versus the false positive rate on the x-axis. The true positive rate (TPR) is the recall and the false positive rate (FPR) is the proportion of all negatives that still yield positive outcomes.

$$\boldsymbol{FPR} = \text{False Positives}/(\text{False Positives} + \text{True Negatives}) \tag{5.3}$$

Additionally, the performance of the recommender system is judged by Hit Ratio (HR) . Hit Ratio is one of the most commonly used performance evaluation metric for ranking based recommender systems. Hit Ratio (HR) is defined as:

$$\boldsymbol{HR} = \frac{\text{number of hits}}{\text{number of users}} \tag{5.4}$$

HR is evaluated at different recommendation lengths ranging from 10 to 100 with certain intervals. It is denoted as HR@n where n is length of the recommendation list. The number of hits represents the hotels present in the test set that are also

recommended by the model and the total number of users gives the number of people in the test set. To generate recommendation list for each user, we combined each user with the all the hotels and evaluated the likelihood of each user is going to the hotel and truncated the highest ranked $\boldsymbol{n}$ as recommendation list. HR ranges from 0, which denotes that the algorithm was not able to recommend anything to 1, which denotes that algorithm is able to perfectly make recommendations. One drawback of HR is that it does not regard where the hit appears in the ranked list. Also, it is not informative. Since hotel could have been prefferred by the user if recommended.

5.2 Baselines

We have compared our method with three sampling methods widely used by recommender systems; random sampling, all missing as negative and weighted version of all missing as negative.

Random Sampling is widely used to tailor negative instances to evaluate performance of the recommender systems. There are two common approaches for the optimization of the learning algorithm in the absence of negative samples. Either treating the entire set of unobserved values as negatives or randomly sub-sampling from this set. In the latter approach we are missing the pairs which would indeed be negative. On the other hand, counting the entire set as negative is a sub-optimal strategy because we are losing the user-hotel pairs which may likely turn to be positive in the future. Also, calling the entire set as negative causes a severe class imbalance problem. Furthermore, it is not necessary to learn model on all entries. **Weighted-All Missing as Negative**: The main idea behind this is if a user has viewed more items, those items that she has not viewed could be negative with higher probability. In weighted regularized matrix factorization, [Y. Hu and Volinsky, 2008] presented a matrix factorization based method to predict items from implicit feedback. Their optimization criteria and learning method slightly differs from our approach. They use SVD to minimize square-loss and weights in the error function to increase the

impact of the positive feedback. The optimization criterion can be summarized as:

$$\sum_{u \in U} \sum_{i \in I} \boldsymbol{c_{ui}}(\langle \boldsymbol{w_u}, \boldsymbol{h_i} \rangle - \mathbf{1})^2 \; + \boldsymbol{\lambda} \|\boldsymbol{W}\|_f^2 + \boldsymbol{\lambda} \|\boldsymbol{H}\|_f^2 \tag{5.5}$$

where $\boldsymbol{c_{ui}}$ represents the a priori given weights for each user and item pair. [Y. Hu and Volinsky, 2008] suggest to set $\boldsymbol{c_{ui}} = \mathbf{1}$ for positive feedback and choose lower constants for the rest. We followed the same idea and the results showed that using weighting to predict how likely an instance is negative performs poorly because the hotels are not the products that are purchased so often and data is quite sparse in this case. Giving higher weights to positives would increase recall but drop the overall performance during prediction(AUC).

5.3 Results

In Chapter 4, a novel negative sampling method has proposed to learn negative examples by using a related task to the recommendation. In this section, the performance evaluation results and the comparison with the learning methods will be discussed in detail.

5.3.1 Baseline Comparison

The results presented in Table 5.1 summarizes the results of the methods discussed in Section 5.2 and Related Task Sampling. We use the following metrics to compare the methods; Precision, Recall, Area Under the Receiver Operating Characteristic Curve(AUC) and Hit Ratio at several thresholds.

We first analyze the results of sub-sampling methods. Where a sample negative dataset the similar size to the training dataset is randomly sampled from all the missing instances and compared the performance with our sampling method. Figure 5.1 showed that randomly sampling performed poorly in overall prediction. Then, we applied the idea of weighting these negative instances. According to the results shown in Table 5.1 the recall value for the weighting based approach is 0.83 bigger than the random sampling recall value 0.77 because giving 1.0 as constant weight to

the positives and smaller constant to the sampled negative instances dominates the importance of positive values. Since the positive values are dominated, it is expected to see that recall has increased. When we compared the weighting based approach recall against our proposed method we observed that our approach is better to predict the relevant items. Also from the table when we compare the overall performance of the recommender system by AUC, it is clear that related task sampling performs better than the baselines in prediction quality.

Table 5.1: Comparison of baseline method with Related Task Sampling

	Precision	Recall	AUROC
Random Sampling with NN	0.880 ± 0.026	0.773 ± 0.033	0.915 ± 0.004
Weighting based Sampling with NN	0.780 ± 0.004	0.830 ± 0.006	0.890 ± 0.009
Related Task Sampling with NN	0.974 ± 0.008	0.860 ± 0.017	0.951 ± 0.002
Curriculum Sampling	0.975 ± 0.006	0.856 ± 0.018	0.950 ± 0.002
BinaryClassifier SVM with Random Sampling	0.760	0.610	0.760
BinaryClassifier SVM with Related Task Sampling	0.823	0.828	0.825
One-Class Classification	-	-	0.56
Matrix Factorization	-	-	0.55

As described earlier 2.4.1, the matrix factorization assumes all the missing entries as unknown and proposes to approximate the missing rating values by only using the positive examples. As expected, since it only uses a small set of positive examples, it fails at finding similar users, Table 5.2. The experiments with a real-world dataset conducted on two similar approaches; one class classification(One Class SVM) and matrix factorization which rely on the availability of explicit feedback performed just slightly better than random assumption. Where AUC values are 0.56 and 0.55 respectively Table 5.1. Also, by adding side information to the matrix factorization we showed that additional features does not really have impact on performance of predictions in our dataset as shown in Table 5.2. This is likely due to the sparse nature of our data.

Table 5.2: Matrix Factorization (MF) Results

	AUROC	RMSE
MF without side information	0.557	0.84
MF with only user features	0.567	0.84
MF with only hotel features	0.578	1.00
MF with user and hotel features	0.568	0.85

We can clearly see that the best results lie in between. That is to say related task sampling outperforms the baselines.

Although hit ratio is a ranking based evaluation metric in Table 5.3 we showed the hit ratio results at several recommendation lists with different sizes. From the table we could see that when we expected to the see the test hotels in the highest ranked 10 in recommendation list we perform poorly than the other approaches. Because neural network is not optimized to rank the items. It is mostly likely that the predicted positive instance can be anywhere. While increasing the size of recommendation list we performed better than the baselines which supported the results that our proposed method performs better in overall prediction.

Table 5.3: HR@N Comparison of baseline method with Related Task Sampling

	$\boldsymbol{N = 10}$	$\boldsymbol{N = 20}$	$\boldsymbol{N = 50}$	$\boldsymbol{N = 100}$
Random Sampling with NN	0.071	0.114	0.239	0.339
Weighting based Sampling with NN	0.079	0.132	0.238	0.355
Related Task Sampling with NN	0.057	0.130	0.273	0.403

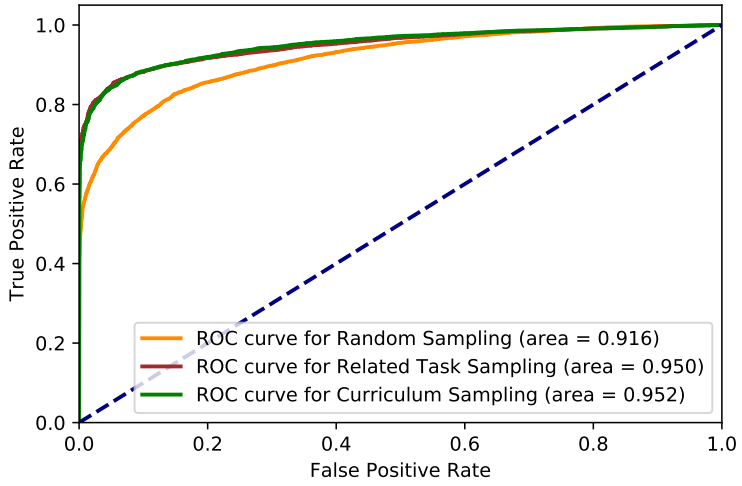

Figure 5.1: Comparison of ROC curves

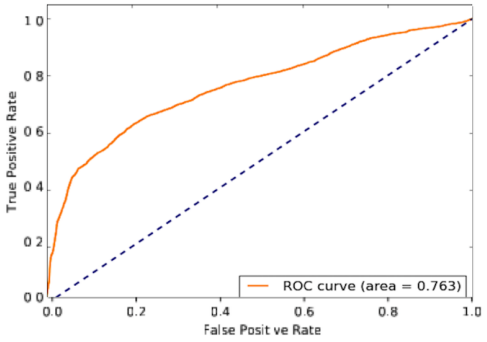

Figure 5.2: 2-Class SVM Random Sampling

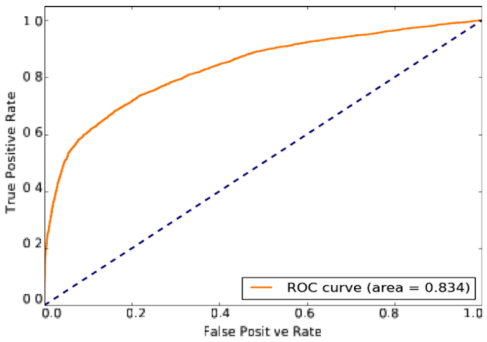

Figure 5.3: 2-Class SVM Related Task Sampling

Chapter 6

CONCLUSION AND FUTURE WORK

Recommender systems face multiple challenges. Inavailability of explicit data is one of them. Most existing systems use implicit data, such as user behavior history. However, it has been shown that explicit data results in better recommendations. Methods for both types of data, such as Matrix Factorization and its variants, have come a long way. However, there is also another challenge that has not been addressed yet; lack of enough user-item pairs for large or low activity domains; namely data sparsity. In this thesis, we address both the implicit data and sparse data problem, specifically for hotel recommendation problem. The main issue arising in this problem is data sparsity, since there are a lot of customers only purchase a few hotels among tens of thousands.

In order to help solve the hotel recommendation problem, we propose two complimentary methods. The first method is to generate missing negative samples to compliment the available purchase history data. Towards this end, we use a related task, namely Click Trough Rate (CTR), to train a deep neural network that takes user and hotel information as data and outputs likelihoods of user clicking. We train this network with real-world data. Then, we generate previously unseen user-hotel pairs and feed it to this network. We take the low likelihood examples as negative data. Then we develop another deep neural network that again takes user and hotel data as input and outputs the purchase probability of the user. We use the existing real world positive labels along with the generated negative labels to train this model.

We evaluate both of our methods to show that:

- Methods that only rely on implicit data fails to perform for our task.

- Our negative data generation approach performs better than existing negative sampling approaches for multiple machine learning models.

- Our deep neural network model for user purchase probability, performs better than other commonly used machine learning models.

However, the performance of our approach is slightly lower for the hit ratio metric for low recommendation sizes. This suggest up an interesting avenue for future work. We want to be able to better rank the recommendations, specifically for low sizes. We may change our network architectures or augment them with another approach for this.

Matrix Factorization variants have been shown to be extremely powerful recommendation approaches. However, their efficacy for sparse data is lacking. Another interesting research direction is to be able to incorporate our generated negative samples into a Matrix Factorization based methods.

Although a set of content features was used in this thesis for training the recommender systems, there might exists many other (content) features on users or hotels that are worth to examine. One example is using hotel images with textual information. This, can unveil what particular characteristics in hotels attract different users. Such system with more concrete information that can be used to provide better targeted recommendations.

Additionally, our approach is not limited to recommender systems or just purchase history-click through rate related tasks. We want to apply our ideas to other related-datasets and machine learning problems in the future.

We close with a word on real-world business deployment scenario based on our experience. Business evaluation of a recommender system is somewhat different from measures like precision and recall. Practical considerations such as expected revenue and time taken to close a transaction, as well as the non-obviousness are also important business factors in judging the value of a recommendation.

BIBLIOGRAPHY

[Adomavicius and Tuzhilin, 2005] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749.

[Aggarwal, 2016] Aggarwal, C. C. (2016). Knowledge-based recommender systems. In *Recommender Systems*, pages 167–197. Marcel Dekker.

[Aghdam et al., 2017] Aghdam, M. H., Analoui, M., and Kabiri, P. (2017). Collaborative filtering using non-negative matrix factorisation. *J. Inf. Sci.*, 43(4):567–579.

[Agichtein et al., 2006] Agichtein, E., Brill, E., and Dumais, S. (2006). Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 19–26, New York, NY, USA. ACM.

[Al-Shamri, 2016] Al-Shamri, M. Y. H. (2016). User profiling approaches for demographic recommender systems. *Know.-Based Syst.*, 100(C):175–187.

[Alper and gdc, 2012] Alper, M. E. and gdc, S. G. (2012). Personalized recommendation in folksonomies using a joint probabilistic model of users, resources and tags. In *2012 11th International Conference on Machine Learning and Applications*, volume 1, pages 368–373.

[Amatriain et al., 2009] Amatriain, X., M. Pujol, J., and Oliver, N. (2009). I like it... i like it not: Evaluating user ratings noise in recommender systems. *G. Houben, G. McCalla, F. Pianesi and M. Zancanaro, User Modeling, Adaptation, and Personalization. Springer Berlin Heidelberg, Berlin, Heidelberg*, 42(34):247–258.

[Balabanovíc and Shoham, 1997] Balabanovíc, M. and Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72.

[Bansal et al., 2016] Bansal, T., Belanger, D., and McCallum, A. (2016). Ask the gru: Multi-task learning for deep text recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 107–114, New York, NY, USA. ACM.

[Batista et al., 2004] Batista, G. E. A. P. A., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20–29.

[Bennett et al., 2007] Bennett, J., Lanning, S., and Netflix, N. (2007). The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*.

[Betru et al., 2017] Betru, B. T., Onana, C. A., and Batchakui, B. (2017). Deep learning for recommender system: Principles, methods and evaluation. 11(9):1028.

[Breazeal, 2003] Breazeal, C. (2003). Toward sociable robots. *Robotics and Autonomous Systems*, 42(34):167 – 175.

[Burke, 2000] Burke, R. (2000). Knowledge-based recommender systems. In *ENCYCLOPEDIA OF LIBRARY AND INFORMATION SYSTEMS*, page 2000. Marcel Dekker.

[Burke, 2002] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370.

[Cantador and Cremonesi, 2014] Cantador, I. and Cremonesi, P. (2014). Tutorial on cross-domain recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 401–402, New York, NY, USA. ACM.

[Chawla et al., 2004] Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Editorial: Special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6(1):1–6.

[Chen et al., 2016] Chen, C.-M., Tsai, M.-F., Lin, Y.-C., and Yang, Y.-H. (2016). Query-based music recommendations via preference embedding. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 79–82, New York, NY, USA. ACM.

[Chen et al., 2017a] Chen, X., Zhang, Y., Ai, Q., Xu, H., Yan, J., and Qin, Z. (2017a). Personalized key frame recommendation. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 315–324, New York, NY, USA. ACM.

[Chen et al., 2017b] Chen, Y., Zhao, X., and de Rijke, M. (2017b). Top-n recommendation with high-dimensional side information via locality preserving projection. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 985–988, New York, NY, USA. ACM.

[Chih-Wei et al., 2003] Chih-Wei, H., Chih-Chung, C., and Chih-Jen, L. (2003). A practical guide to support vector classification.

[Chu and Tsai, 2017] Chu, W.-T. and Tsai, Y.-L. (2017). A hybrid recommendation system considering visual information for predicting favorite restaurants. *World Wide Web*, 20(6):1313–1331.

[Claypool et al., 1999] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., and Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *In Proceedings of ACM SIGIR Workshop on Recommender Systems*.

[Covington et al., 2016] Covington, P., Adams, J., and Sargin, E. (2016). Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 191–198, New York, NY, USA. ACM.

[Cristianini and Shawe-Taylor, 2000] Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines: And Other Kernel-based Learning Methods*. Cambridge University Press, New York, NY, USA.

[Das et al., 2007] Das, A., Datar, M., Rajaram, S., and Garg, A. (2007). Google news personalization: scalable online collaborative filtering. In *in WWW, 2007*, pages 271–280.

[Deshpande and Karypis, 2004] Deshpande, M. and Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177.

[Dong et al., 2017] Dong, X., Yu, L., Wu, Z., Sun, Y., Yuan, L., and Zhang, F. (2017). A hybrid collaborative filtering model with deep structure for recommender systems. In *AAAI*.

[Drummond and Holte, 2003] Drummond, C. and Holte, R. C. (2003). C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. pages 1–8.

[Elkahky et al., 2015] Elkahky, A. M., Song, Y., and He, X. (2015). A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 278–288, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

[Elkan and Noto, 2008] Elkan, C. and Noto, K. (2008). Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD Inter-*

national Conference on Knowledge Discovery and Data Mining, KDD '08, pages 213–220, New York, NY, USA. ACM.

[Fang and Si, 2011] Fang, Y. and Si, L. (2011). Matrix co-factorization for recommendation with rich side information and implicit feedback. In *HetRec '11*.

[Gabriel and Zamir, 1979] Gabriel, K. R. and Zamir, S. (1979). Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics*, 21(4):489–498.

[Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy. PMLR.

[Gunawardana and Shani, 2009] Gunawardana, A. and Shani, G. (2009). A survey of accuracy evaluation metrics of recommendation tasks. *J. Mach. Learn. Res.*, 10:2935–2962.

[Guo and Berkhahn, 2016] Guo, C. and Berkhahn, F. (2016). Entity embeddings of categorical variables. *CoRR*.

[Guo et al., 2017] Guo, H., Tang, R., Ye, Y., Li, Z., and He, X. (2017). Deepfm: A factorization-machine based neural network for CTR prediction. *CoRR*, abs/1703.04247.

[He et al., 2017] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 173–182, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

[Herlocker et al., 2004] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53.

[Hu et al., 2008] Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA. IEEE Computer Society.

[Huang et al., 2007] Huang, Z., Zeng, D., and Chen, H. (2007). A comparison of collaborative-filtering recommendation algorithms for e-commerce. *IEEE Intelligent Systems*, 22(5):68–78.

[Iyengar and Lepper, 2000] Iyengar, S. and Lepper, M. (2000). When choice is demotivating: Can one desire too much of a good thing? *Journal of Personality and Social Psychology*, 79(6):995–1006.

[Jiang et al., 2012] Jiang, M., Cui, P., Wang, F., Yang, Q., Zhu, W., and Yang, S. (2012). Social recommendation across multiple relational domains. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1422–1431, New York, NY, USA. ACM.

[Joachims, 2002] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, New York, NY, USA. ACM.

[Karatzoglou and Hidasi, 2017] Karatzoglou, A. and Hidasi, B. (2017). Deep learning for recommender systems. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 396–397, New York, NY, USA. ACM.

[Katarya and Verma, 2015] Katarya, R. and Verma, O. P. (2015). Restaurant recommender system based on psychographic and demographic factors in mobile environment. In *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)(ICGCIOT)*, volume 00, pages 907–912.

[Katarya and Verma, 2016] Katarya, R. and Verma, O. P. (2016). Recommender system with grey wolf optimizer and fcm. *Neural Computing and Applications*, 30:1679–1687.

[Kelly and Teevan, 2003] Kelly, D. and Teevan, J. (2003). Implicit feedback for inferring user preference: A bibliography. *SIGIR Forum*, 37(2):18–28.

[Khan et al., 2017] Khan, M. M., Ibrahim, R., and Ghani, I. (2017). Cross domain recommender systems: A systematic literature review. *ACM Comput. Surv.*, 50(3):36:1–36:34.

[Kim et al., 2017] Kim, D., Park, C., Oh, J., and Yu, H. (2017). Deep hybrid recommender systems via exploiting document context and statistics of items. *Inf. Sci.*, 417(C):72–87.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

[Koenigstein and Paquet, 2013] Koenigstein, N. and Paquet, U. (2013). Xbox movies recommendations: Variational bayes matrix factorization with embedded feature selection. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 129–136, New York, NY, USA. ACM.

[Koren, 2008] Koren, Y. (2008). Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 426–434, New York, NY, USA. ACM.

[Koren et al., 2009] Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

[Li and Chen, 2015] Li, G. and Chen, Q. (2015). Exploiting explicit and implicit feedback for personalized ranking. *Mathematical Problems in Engineering*, 2016(Gai Li1 and Qiang Chen2).

[Li et al., 2010] Li, Y., Hu, J., Zhai, C. X., and Chen, Y. (2010). Improving one-class collaborative filtering by incorporating rich user information. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 959–968, New York, NY, USA. ACM.

[Linden et al., 2003] Linden, G., Smith, B., and York, J. (2003). Amazon.Com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1):76–80.

[Liu et al., 2003] Liu, B., Dai, Y., Li, X., Lee, W. S., and Yu, P. S. (2003). Building text classifiers using positive and unlabeled examples. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 179–, Washington, DC, USA. IEEE Computer Society.

[Liu et al., 2006] Liu, X.-Y., Wu, J., and Zhou, Z.-H. (2006). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39:539–550.

[Marlin et al., 2012] Marlin, B. M., Zemel, R. S., Roweis, S. T., and Slaney, M. (2012). Collaborative filtering and the missing at random assumption. *CoRR*, abs/1206.5267.

[Massa and Avesani, 2007] Massa, P. and Avesani, P. (2007). Trust-aware recommender systems. In *Proceedings of the 2007 ACM Conference on Recommender Systems*, RecSys '07, pages 17–24, New York, NY, USA. ACM.

[Musto et al., 2016] Musto, C., Greco, C., Suglia, A., and Semeraro, G. (2016). Ask me any rating: A content-based recommender system based on recurrent neural networks. In *IIR*.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

[Nati and Jaakkola, 2003] Nati, N. S. and Jaakkola, T. (2003). Weighted low-rank approximations. In *In 20th International Conference on Machine Learning*, pages 720–727. AAAI Press.

[Ning and Karypis, 2010] Ning, X. and Karypis, G. (2010). Multi-task learning for recommender system. In Sugiyama, M. and Yang, Q., editors, *Proceedings of 2nd Asian Conference on Machine Learning*, volume 13 of *Proceedings of Machine Learning Research*, pages 269–284, Tokyo, Japan. PMLR.

[Pan and Scholz, 2009] Pan, R. and Scholz, M. (2009). Mind the gaps: Weighting the unknown in large-scale one-class collaborative filtering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 667–676, New York, NY, USA. ACM.

[Pan et al., 2008] Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., and Yang, Q. (2008). One-class collaborative filtering. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 502–511, Washington, DC, USA. IEEE Computer Society.

[Paquet and Koenigstein, 2013] Paquet, U. and Koenigstein, N. (2013). One-class collaborative filtering with random graphs. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 999–1008, New York, NY, USA. ACM.

[Popescul et al., 2013] Popescul, A., Ungar, L. H., Pennock, D. M., and Lawrence, S. (2013). Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. *CoRR*, abs/1301.2303.

[Rawat and Kankanhalli, 2016] Rawat, Y. S. and Kankanhalli, M. S. (2016). Contagnet: Exploiting user context for image tag recommendation. In *Proceedings of the 2016 ACM on Multimedia Conference*, MM '16, pages 1102–1106, New York, NY, USA. ACM.

[Rendle et al., 2009] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2009). Bpr:bayesian personalized ranking from implicit feedback. *UAI*, 42(34):452461.

[Rendle et al., 2011] Rendle, S., Gantner, Z., Freudenthaler, C., and Schmidt-Thieme, L. (2011). Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 635–644, New York, NY, USA. ACM.

[Salakhutdinov et al., 2007] Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 791–798, New York, NY, USA. ACM.

[Sedhain et al., 2015] Sedhain, S., Menon, A. K., Sanner, S., and Xie, L. (2015). Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, pages 111–112, New York, NY, USA. ACM.

[Shen et al., 2005] Shen, X., Tan, B., and Zhai, C. X. (2005). Implicit user modeling for personalized search. In *Proceedings of the 14th ACM International Conference*

on Information and Knowledge Management, CIKM '05, pages 824–831, New York, NY, USA. ACM.

[Shuai Zhang, 2017] Shuai Zhang, Lina Yao, a. A. S. (2017). Toward sociable robots. *arXiv preprint arXiv:1707.07435*, 42(34).

[Sindhwani et al., 2009] Sindhwani, V., Bucak, S. S., Hu, J., and Mojsilovic, A. (2009). A family of non-negative matrix factorizations for one-class collaborative filtering problems.

[Sindhwani et al., 2010] Sindhwani, V., Bucak, S. S., Hu, J., and Mojsilovic, A. (2010). One-class matrix completion with low-density factorizations. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 1055–1060, Washington, DC, USA. IEEE Computer Society.

[Smola and Schölkopf, 2004] Smola, A. J. and Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222.

[Verstrepen et al., 2017] Verstrepen, K., Bhaduriy, K., Cule, B., and Goethals, B. (2017). Collaborative filtering for binary, positive only data. *SIGKDD Explor. Newsl.*, 19(1):1–21.

[Wang et al., 2016] Wang, C., Liu, Y., and Ma, S. (2016). Building a click model: From idea to practice. *CAAI Transactions on Intelligence Technology*, 1(4):313 – 322.

[Wang et al., 2012] Wang, Y., Chan, S. C., and Ngai, G. (2012). Applicability of demographic recommender system to tourist attractions: A case study on trip advisor. In *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 97–101.

[Weston et al., 2011] Weston, J., Bengio, S., and Usunier, N. (2011). Wsabie: Scaling up to large vocabulary image annotation. In *Proceedings of the Twenty-Second*

International Joint Conference on Artificial Intelligence - Volume Volume Three, IJCAI'11, pages 2764–2770. AAAI Press.

[Weston et al., 2013] Weston, J., Yee, H., and Weiss, R. (2013). Learning to rank recommendations with the k-order statistic loss. In *ACM International Conference on Recommender Systems (RecSys)*.

[Y. Hu and Volinsky, 2008] Y. Hu, Y. K. and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. *ICDM*, 42(34):263–272.

[Yoshua et al., 2009] Yoshua, B., Jérôme, Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA. ACM.

[Yu, 2003] Yu, H. (2003). Svmc: Single-class classification with support vector machines. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 567–572, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Yu et al., 2017] Yu, H.-F., Bilenko, M., and Lin, C.-J. (2017). Selection of negative samples for one-class matrix factorization. In *Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, Texas, USA, April 27-29, 2017.*, pages 363–371.

[Yu and Huang, 2017] Yu, H.-F. and Huang, H.-Y. (2017). A unified algorithm for one-class structured matrix factorization with side information.

[Yu and Lin, 2016] Yu, H.-F.; Bilenko, M. and Lin, C.-J. (2016). Selection of negative samples for one-class matrix factorization. *SDM*, 42(34):452461.

[Zhang et al., 2017] Zhang, S., Yao, L., and Sun, A. (2017). Deep learning based recommender system: A survey and new perspectives. *CoRR*, abs/1707.07435.

[Zhang et al., 2013] Zhang, W., Chen, T., Wang, J., and Yu, Y. (2013). Optimizing top-n collaborative filtering via dynamic negative item sampling. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 785–788, New York, NY, USA. ACM.

[Zhang and Jiao, 2007] Zhang, Y. and Jiao, J. R. (2007). An associative classification-based recommendation system for personalization in b2c e-commerce applications. *Expert Syst. Appl.*, 33(2):357–367.

[Zhao et al., 2016] Zhao, F., Xiao, M., and Guo, Y. (2016). Predictive collaborative filtering with side information. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 2385–2390. AAAI Press.

[Zhao et al., 2015] Zhao, W. X., Li, S., He, Y., Wang, L., Wen, J.-R., and Li, X. (2015). Exploring demographic information in social media for product recommendation. *Knowledge and Information Systems*, 49:61–89.

[Zhou et al., 2008] Zhou, Y., Wilkinson, D., Schreiber, R., and Pan, R. (2008). Large-scale parallel collaborative filtering for the netflix prize. In *Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management*, AAIM '08, pages 337–348, Berlin, Heidelberg. Springer-Verlag.