Insert here your thesis' task.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Master's thesis

Algorithms for collaborative filtering in Point-of-Interest Recommendation Systems

Bc. Guzel Samigullina

Department of Software Engineering
Supervisor: Ing. Jaroslav Kuchář, Ph.D.

May 7, 2019

Acknowledgements

I would like to thank my supervisor Ing. Jaroslav Kuchář, Ph.D. for all recommendations and insights he made while writing the thesis. Also, I would like to thank all who supported me during the work on this thesis, especially my family and friends.

Declaration

In Prague on May 7, 2019 . . . .

Citation of this thesis

Samigullina, Guzel. *Algorithms for collaborative filtering in Point-of-Interest Recommendation Systems.* Master's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2019.

Abstrakt

S dostupností obrovského množství uživatelů a geolokačních sociálních sítí získal v posledních letech problém doporučení míst zájmu značnou pozornost výzkumu. Zatímco předchozí práce zabývající se doporučením míst zájmu se většinou zaměřovaly na zkoumání prostorového, časového a sociálního vlivu, použití dodatečných obsahových informací nebylo cíleně studováno. Tyto dodatečné informace mohou nejen zlepšit kvalitu doporučení, ale také překonat i problém tzv. studeného startu.

V této práci navrhuji algoritmus pro doporučení míst zájmu založený na faktorizaci matice s přidanou obsahovou informací – atributy a kategorie míst zájmu. Navrhuji dvě varianty algoritmu, které mohou pracovat s explicitní a implicitní zpětnou vazbou. Informace o atributech a kategoriích jsou získány z existujících datových sad a použity k měření podobnosti mezi dvěma místy. Experimentální výsledky ukazují, že navrhovaná metoda zlepšuje kvalitu doporučení, překonává většinu populárních algoritmů kolaborativního filtrováni a dokáže efektivně zvládnout problém studeného startu.

Klíčová slova Systém doporučování míst zájmu, faktorizace matice, vážená maticová faktorizace, geolokační sociální síť, SGD, ALS

Abstract

With the availability of the vast amount of users and Location-based social networks, the problem of POI recommendations has been widely studied and received significant research attention in the last seven years, and many approaches have been suggested. While previous works of POI recommendation mostly focused on investigating the spatial, temporal, and social influence, the use of additional content information has not been directionally studied. Such additional information can not only improve the performance of the recommendation system but also help to overcome the so-called “cold start” problem.

In this paper, we propose the content-aware matrix factorization method based on incorporating POI attribute and categories information. We propose two variants of the algorithm that can work with explicit and implicit feedback. The attribute and categories information of a POI is collected from existing datasets and used to measure the similarity between two POIs. These similarity values are subsequently used as a regularization term added to the objective function of matrix factorization. Experimental results show that the proposed method improves the quality of recommendation, outperforms most state-of-the-art collaborative filtering algorithms and can effectively cope with the so-called “cold start” problem.

Keywords POI Recommendation System, matrix factorization, weighted matrix factorization, Location-based Social Network, SGD, ALS

Contents

List of Figures

Introduction

Location-based social networks (LBSNs) have become very popular and attracted lots of attention from internet users, business and academia with the increasing popularity of GPS-enabled mobile devices. In LBSNs, users can build connections with their friends, upload photos, and share their locations by check in the points of interest (e.g., restaurants, famous landmarks, housing locations, and parking spaces) [1]. Typical location-based social networks include Foursquare, Yelp, Facebook Place, GeoLife, etc. The number of users in those networks is huge, for example, Foursquare had more than 50 million monthly active users on October 2018 [2], and Yelp had about 33 million unique desktop visitors and 69 million mobile visitors by the end of 2018 [3].

In the meantime, making a satisfying decision among a large number of point of interests (POI) becomes a difficult problem for a user, as well known as a “choice paralysis”. POI recommendation is a task that aims to address such issue by helping users filter out uninteresting POIs and reduce their decision-making time. Also, POI recommender systems have played an essential role in LBSNs, because they help to increase revenues by providing users with intelligent location services, such as location-aware advertisements [1].

With the availability of the vast amount of users’ visiting history, the problem of POI recommendations has been widely studied and received significant research attention in the last seven years, and many approaches have been suggested. While previous works of POI recommendation mostly focused on investigating the spatial, temporal, and social influence, the use of additional content information has not been directionally studied. Such additional information can not only improve the performance of the recommendation system but also help to overcome the so-called cold start problem.

In this paper, we propose the content-aware matrix factorization method based on incorporating POI attribute and categories information to overcome the cold start item problem, and consequently improve the quality of recommendation. We propose two variants of the algorithm that can work with explicit and implicit feedback. The attribute and categories information of a

POI will be collected from existing datasets and will be used to measure the similarity between two POIs. This similarity values will be used to regularize the matrix factorization by adding item relationship regularization term to the objective function of matrix factorization [4].

This work is organized as follows. First chapter gives a brief introduction of recommendation systems (Section 1.1), describes in detail memory-based (Section 1.2) and model-based (Section 1.3) collaborative filtering methods, describes unique characteristics of POI recommendation systems in location-based social networks (Section 1.4) and reviews related research directions for POIs recommendation in Section 1.5. Chapter 2 describes the details of the used content-aware POI recommendation algorithm. Chapter 3 defines the properties of existing datasets and ways of preprocessing and joining datasets. In Chapter 4 experiments are evaluated and important implementation details are described. Finally, at the end of paper conclusion and some directions for future work are presented.

Chapter 1

Recommendation Systems

1.1 An Introduction to Recommendation Systems

Recommendation systems are intelligent software tools that helps a user discover products and content by predicting the user's rating of each item and providing with the decision making support information, such as what book to read next, what movie to watch and what new place to visit. At present, recommendation systems have become indispensable since it helps to cope with the information overload problem, which is especially aggravated with the rapid growth of internet users and active development of smart devices.

Recommendation systems are based on users feedback that specifies their likes and dislikes of various items. In traditional recommendation systems, user generally expressed their preferences by explicitly providing ratings for items (e.g., movie, book, place and so on), typically on a concrete rating scale (e.g., five-star rating system). This type of feedback is called **explicit** rating. Other forms of feedback are not quite as explicit but are even easier to collect thanks to the global spread of the internet. Example of such **implicit** feedback can be the simple actions of a user, such as the number of clicks on this product, the frequency of check-ins in some location, the number of times a soundtrack is played and so on.

As soon as user preferences are collected, the recommendation problem is reduced to the problem of estimating ratings for the items that have not been seen by a user. According to the approach of rating estimation, recommendation systems are usually classified into four basic models, that will be briefly discussed in the following subsections.

1.1.1 Collaborative Filtering Systems

Collaborative filtering systems collect information of the user-item interactions, such as implicit or explicit ratings, onto rating matrix and try to predict the utility of items for a particular user based on the items previously rated by other users [5].

According to [6], algorithms for collaborative recommendations can be grouped into two general classes: memory-based and model-based:

1. *Memory-based algorithms*: These systems employ statistical techniques to find a set of users, known as neighbors, that have similar patterns of rating behavior with the target user (i.e., they either rate different items similarly or they tend to buy similar sets of items) [7], or set of neighbor items, that are similar to user's already-rated items. These neighborhoods can be defined in one of two ways:

   - *User-based collaborative filtering*: The basic idea of this method is to predict the ratings based on the preferences of active user's neighbors, which have similar favorites with an active user. In a book recommendation application, to recommend a book to some user, the user-based collaborative recommendation system tries to find other users that have similar tastes (rate the same book similarly). Then, only the books that are most liked by the k most similar users would be recommended. User-based methods were widely studied, some variations are provided in [8, 9].
   - *Item-based collaborative filtering:* On the other hand, item-based approaches provide predictions based on the ratings given by the active user for items similar to target items. For example, in a book recommendation application, to predict the rating of the target user A for some book B, the most similar books (denoted as S) to book B are detected from the set of the already-rated books by user A. Then, the weighted average of ratings from S is used to calculate the predicted rating for book B. A detailed study of item-based collaborative filtering methods is provided in [10], in [11] is described one of Amazon.com's user-based collaborative filtering algorithms.

   The main advantages of memory-based techniques that they are easy to implement, they are context independent and compared to other methods, such as content-based, it is more accurate. But these systems have some problems, such as rating sparsity, new item problem and cold-start for new users. Memory-based methods are discussed in detail in Section 1.2.

2. *Model-based algorithms:* This approach first makes use of data mining and machine learning techniques to learn a predictive model from training data. That model can characterize the rating behaviors of target users. Then model-based method uses the trained model to make predictions, rather than directly utilize the entire user-item matrix to com-

pute predictions [4]. Typical examples of model-based filtering methods include decision trees [12], association rules [13, 14], Bayesian networks [9], latent semantic models [15] and clustering model [16]. Model-based methods are discussed in detail in Section 1.3.

Generally, memory-based algorithms tend to easy to implement and produce reasonable high prediction quality. Also, the results of the recommendation are often easy to explain. However, memory-based algorithms suffer from a severe scalability problem. Especially this applies to modern E-commerce sites because constant growth of users and items slow down their online performance.

Model-based algorithms tend to be faster than memory-based algorithms because the time required to query the model is usually much smaller than the time required to query the whole dataset. But its disadvantages are that many theoretical models are complex and sometimes are not fit well with real data. Also, it is more difficult and takes a long time to build or update models for model-based algorithms, making them inflexible.

1.1.2 Content-Based Recommendation Systems

Content-Based Recommendation Systems use the content information available in the items and combines it with the ratings and buying behavior for recommending purposes. For example, an active user has rated an item highly, but we do not have access to the ratings of other users. However, we can assign this item to a specific category from the item description. In such cases, items from this category can be recommended to the active user. The content of an item gives us a lot of options. For example, for a movie, we could consider not only the plot description but also the director, the genres, the actors and so on.

According to [6], the whole process of content-based recommendation can be divided into three basic components:

1. *Preprocessing and feature extraction:* After we decided which content we will consider, the next step is to transform all this data into an algebraic representation of text documents, i.e., to keyword-based vector-space representation. Generally, we do this with a Bag of Words model, where each document looks like a bag containing some words without any order [17]. However, before determining the bags of words, data need to be cleaned in the several steps, such as stop-word removing, stemming and lemmatization, and phrase extraction.

2. *Content-based learning of user profiles:* At this step user feedbacks (explicit or implicit) are used in combination with the content information of items to construct the training data. On this training data, a profile learner is constructed, that represents each user's preferences.

3. *Filtering and recommendation:* The learned model from the previous step takes all the inputs and generates the list of recommendations for each user.

The content-based method is trying to solve the problems of the collaborative filtering algorithms. The main advantages of content-based method are:

- Unlike Collaborative Filtering, this method doesn't suffer from ”new item problem” because new items have descriptions and categorization.
- The content-based system is **user independent** because do not use ratings from other users.
- It is easy to provide explanations because the same content is used to explain the recommendations.
- Content representations are varied and they open up the options to use different approaches like: text processing techniques, the use of semantic information, inferences and so on [17].

1.1.3 Knowledge-Based Recommendation Systems

In knowledge-based recommender systems recommendations are suggested on the basis of user-specified requirements rather than the historical rating or buying data of the user. Such recommender systems are particularly useful in the context of items that are not purchased very often or rating history not available for the recommendation process(for example financial services or tourism requests). Knowledge-based methods are unique in that they allow the users to explicitly specify what they want that give users greater control over the recommendation process.

1.1.4 Hybrid Recommendation Systems

All above-mentioned algorithms have different strengths and weaknesses, and each of the methods can be more effective in different cases. For example, collaborative filtering systems depend on user ratings, content-based algorithms rely on textual item descriptions, and knowledge-based methods rely on interactions with the users in the context of knowledge bases.

There are various hybridization strategies that can be applied. According to [5], these strategies are easiest to explain on the most frequently used hybrid recommender system – a combination of collaborative and content-based methods:

- *Ensemble design – combining separate recommender methods:* In this case, collaborative and content-based systems are implemented separately. There are two different combination strategies – weighted and switching model. In weighted hybrids, the outputs (ratings) obtained from individual recommender systems are combined using a set of weights, for example using a voting scheme [18] or linear combination of ratings [19]. Switching mechanisms are often used to handle the cold-start problem, in which one of the individual recommenders is chosen at any given moment because it is "better" than others based on some recommendation quality metric. For example, the Daily Learner system [20] selects the recommender system that can recommend with a higher level of confidence.

- *Monolithic design:* This method of hybridization is divided to **feature combination** and **meta level** strategies. In feature combination hybrids, the idea is to combine the input data from various sources (e.g., content and collaborative) into a unified representation before applying a predictive algorithm [6]. The most popular approach in this category is to add collaborative feature to content-based models. In a meta-level hybrid, one recommender system is used as an input to another system. The typical method described in [18] is referred to as "collaboration via content," where a collaborative system is modified to use content features to determine peer groups.

- *Mixed hybrids:* This approach is most appropriate in complex item domains, and it is often used in combination with knowledge-based recommender systems.

1.2 Memory-based collaborative filtering

Memory-based collaborative filtering methods also referred to as neighborhood-based algorithms, were among the first algorithms developed for collaborative filtering. These systems employ statistical techniques to find similar users or similar items (i.e., "neighborhoods") to make predictions.

This section is organized as follows. Two basic types of memory-based collaborative filtering – user-based and item-based collaborative filtering models are discussed in detail in the following subsections 1.2.1 and 1.2.2. The differences between these two methods are discussed in the subsection 1.2.3. A summary of memory-based methods is given in the subsection 1.2.4.

1.2.1 User-based collaborative filtering

The basic idea of this method is to predict the ratings based on the preferences of active user's neighbors, which have similar favorites with an active user. The most popular application of this method is the user-based Nearest Neighbor algorithm (k-NN), that can be reduced to two steps:

1. Find k most similar users (users, that have similar tastes to target user). An essential part of this step is to choose the appropriate similarity function, that measure the distance between each pair of users. Multiple similarity measures are discussed in the section 1.2.1.1.

2. Predict the ratings that active user will give to all unrated items using the ratings from those k like-minded users found in first step. Variants of prediction functions are discussed in the section 1.2.1.2.

The user-based model works with $m \times n$ user-item ratings matrix, where m is the number of users and n is the number of items. For the purpose of subsequent discussion, the user-item ratings matrix will be denoted by R. Each entry r_{ij} of R represents the rating given by user i for item j. The set of items rated by the user u is denoted as I_u and the set of items rated by both users u and v is specified by $I_u \cap I_v$. In practice, the rating matrix R is generally very sparse with many unknown entries since a typical user may have only rated a tiny percentage of items.

1.2.1.1 Similarity functions

To find like-minded users, similarity functions $Sim(u, v)$ are computed between the rating vectors of users u a v, i.e., between the rows of the rating matrix R. Various similarity functions are used in practice. Choosing an appropriate similarity measure is very important for a recommender system because different similarity measures provide different results in various contexts of the information. Below are described some of the popular similarity measures metrics that are used in collaborative filtering.

Pearson correlation The most commonly used method. This metric is used to find a linear correlation between the two vectors, that measured from -1 to $+1$. Pearson Correlation Coefficient -1 represents a negative correlation and means that the data objects are not correlated, while $+1$ indicates that the data objects are perfectly correlated, i.e., the larger coefficient the more similar users to each other. Zero value shows no relation sometimes called zero-order correlation.

Pearson correlation coefficient is computed between the rating vectors of two users u and v. The first step is to compute the mean rating μ_u for each user u and target user v over the items that are rated both by users u and v [6]:

$$\mu_{uv} = \frac{\sum_{k \in I_u \cap I_v} r_{uk}}{|I_u \cap I_v|} \quad \forall u \in \{1...m\}$$

Then, the Pearson correlation coefficient between the rating vectors of users u a v is defined as follows:

$$Sim(u,v) = PC(u,v) = \frac{\sum_{k\in I_u\cap I_v}(r_{uk}-\mu_{uv})\cdot(r_{vk}-\mu_{vu})}{\sqrt{\sum_{k\in I_u\cap I_v}(r_{uk}-\mu_{uv})^2}\cdot\sqrt{\sum_{k\in I_u\cap I_v}(r_{vk}-\mu_{vu})^2}}$$

It is the traditional definition of the Pearson correlation, that mandates that the average rating of user u should be computed only over the co-rated items with user v. However, it is quite common (and computationally simpler) to compute average rating just once for each user u over all the items rated by user u [6]. That is exactly the way the Adjusted cosine similarity works (described below).

Cosine Similarity This method is also the most commonly used method to measure the similarity between users in recommender systems. It measures cosine of the angle created from the two vectors in the coordinate system and determines how two vectors are related to each other (i.e., the smaller the angle, the more similar the two vectors are). The angle between the two positive-valued vectors is bound between 0° and 90°, so the cosine similarity returns a value between 0 and 1. One important thing to note is the Cosine similarity is a measure of orientation, not magnitude [21]. Two vectors can be oriented in the same direction (and thus have cosine similarity of 1) but have different magnitudes.

Cosine similarity between the rating vectors of users u a v is defined as follows:

$$Cos(u,v) = \frac{\sum_{k\in I_u\cap I_v} r_{uk}\cdot r_{vk}}{\sqrt{\sum_{k\in I_u\cap I_v} r_{uk}^2}\cdot\sqrt{\sum_{k\in I_u\cap I_v} r_{vk}^2}}$$

The main problem of this method is that it does not consider the scenario in which different users may provide ratings on different scales, e.g., one user can rate all items highly, while another user can rate all items negatively. The following method solves this problem by using mean-centered ratings instead of raw ratings.

Adjusted cosine similarity The adjusted cosine similarity overcomes the drawback of the cosine similarity. This approach uses mean-centered rating of a user u for item k, that is defined by subtracting the mean rating from the raw rating r_{uk}:

$$AdjustedCos(u,v) = \frac{\sum_{k\in I_u\cap I_v}(r_{uk}-\mu_u)\cdot(r_{vk}-\mu_v)}{\sqrt{\sum_{k\in I_u\cap I_v}(r_{uk}-\mu_u)^2}\cdot\sqrt{\sum_{k\in I_u\cap I_v}(r_{vk}-\mu_v)^2}},$$

where mean rating is defined as:

$$\mu_u = \frac{\sum_{k\in I_u} r_{uk}}{|I_u|} \quad \forall u \in \{1...m\} \tag{1.1}$$

The difference between the Adjusted cosine similarity and Pearson Correlation is in the calculation of mean rating for user u. First one computes the mean rating of user u for all the items rated by user "u", while the second one considers the mean rating of user "u" for co-rated items.

Jaccard Coefficient The Jaccard coefficient ignores the rating values and only checks whether the user expressed a preference or not. This metric measures the similarity between two users by counting the number of common rated items and dividing by the total number of unique items that either of the users are interested in. So two users will be more similar when they have more common rated items. The Jaccard coefficient ranges between 0 and 1. The result will be zero if two users do not have any similar preference. The equation for Jaccard Coefficient Similarity between users u and v is:

$$Jaccard(u, v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|},$$

where I_u is a set of items rated by user u and I_v is a set of items rated by user v.

There are several constraints while choosing the best algorithm to be used to measure the similarity. For example, the Pearson coefficient algorithm requires a minimum number of items to be greater than 2 to measure the similarity; Jaccard coefficient ignores the rating values and only checks that whether a user expressed a preference or not, that is why it produces a limited number of values which makes the task of user distinction difficult [22].

Before proceeding with the prediction of ratings, the so-called peer group for a target must be defined. The simplest approach is to use the top-k most similar users to the target user as her peer group. However, such an approach might include users that are weakly or negatively correlated with the target user, that can make the prediction inaccurate and erroneous. Therefore, ratings with weak or negative correlations are often filtered out [6].

1.2.1.2 Prediction functions

Once we compute similarity values between users and define the peer group, we can predict the rating for any user-item pair by using some of prediction functions. There are many variants of the prediction functions used in the user-based collaborative filtering. The two commonly used techniques, weighted sum and regression, are described below.

Weighted sum The most commonly used method, also called the mean-centered prediction function. Prediction is computed as a weighted average of the mean-centered rating of an item in the peer group of the most similar users. Then the mean rating of the active user (using equation 1.1) is added to this prediction. The mean-centered rating of a user u for item j is defined by subtracting her mean rating μ_u from the raw rating r_{uj}. The overall neighborhood-based prediction rating of target user u for item j is as follows [6]:

$$\hat{r}_{uj} = \mu_u + \frac{\sum_{v \in K_u(j)} Sim(u,v) \cdot (r_{vj} - \mu_v)}{\sum_{v \in K_u(j)} |Sim(u,v)|}, \tag{1.2}$$

where $K_u(j)$ is the k most similar users to target user u, who have specified ratings for item j.

Some variation of this function is to use the Z-score z_{uj} instead of mean-centering rating, that is defined as follows [6]:

$$z_{uj} = \frac{r_{uj} - \mu_u}{\sigma_u}, \quad where\ \sigma_u = \sqrt{\frac{\sum_{j \in I_u} (r_{uj} - \mu_u)^2}{|I_u| - 1}}$$

In this case, the predicted rating $\hat{r}_{uj}$ of target user u for item j is defined as follows:

$$\hat{r}_{uj} = \mu_u + \sigma_u \cdot \frac{\sum_{v \in K_u(j)} Sim(u,v) \cdot z_{vj}}{\sum_{v \in K_u(j)} |Sim(u,v)|}$$

In this context, the weighted average needs to be multiplied with σ_u for normalization purposes. It is worth noting that the Z-score has a problem – the predicted ratings might be outside the range of the allowable ratings. Nevertheless, even when the predicted values are outside the range, they can be used to rank the items in order of desirability for a particular user [6].

These methods are variants of linear regression models, in which the regression coefficients are heuristically set to similarity values for neighboring users and to 0 for unrelated users.

Regression Another approach that has been recently widely used is the regression model. The aforementioned **weighted sum** method can also be included in heuristic variants of *linear regression models*, in which the regression coefficients are heuristically set to similarity values for the peer group of target user. Unfortunately, using similarities as heuristic weights do not account for interdependencies among items and may be deceptive in the sense that two rating vectors may be distant but may have very high similarity (for example hight ratings on the same type of movie). That may result in a poor prediction.

This problem can be solved by learning the weights with the use of an optimization formulation of the neighborhood model. The advantage of such models is that the weights for combining the ratings can be better justified because of their optimality from a modeling perspective [6]. Such regression-based model can be defined from Equation 1.2 by replacing the normalized similarity coefficient with the unknown parameter w_{uv}:

$$\hat{r}_{uj} = \mu_u + \sum_{v \in K_u(j)} w_{uv} \cdot (r_{vj} - \mu_v) \tag{1.3}$$

In regression-based model, to define the set $K_u(j)$ first the k closest peers for each user are determined, and then only those ones for which ratings are observed are retained. It is an important difference between the weighted sum method.

To calculate unknown parameter w_{uv}, we can use the aggregate squared difference between the predicted ratings $\hat{r}_{uj}$ and the existing ratings r_{uj} to create the least-squares objective function that estimates the quality of prediction. The objective function for the user u can be defined as the sum of squared errors of prediction and must be minimized:

$$\min \sum_{j \in I_u} (r_{uj} - \hat{r}_{uj})^2) = \min \sum_{j \in I_u} \left(r_{uj} - \left[\mu_u + \sum_{v \in K_u(j)} w_{uv} \cdot (r_{vj} - \mu_v) \right] \right)^2,$$

where I_u is the set of item rated by user u, $K_u(j)$ is the set of the k closest peers for user u. According to [6], the second relationship is obtained by substituting the expression in Equation 1.3. This least-squares optimization problem can be solved using any off-the-shelf optimization solvers, for example, a gradient descent approach can be used. For better prediction regression models can be combined with other optimization models, such as matrix factorization. Such methods are discussed in Section 1.3.

1.2.2 Item-based collaborative filtering

Instead of a user-based approach, item-based collaborative filtering methods provide predictions for items similar to target items based on the ratings given by the active user. In this case similarity functions are computed between the columns of the ratings matrix R to find similar items.

Because item-based algorithms are very similar to user-based algorithms, similar variants of the similarity functions can be considered. The only difference is that similarity functions are computed between two items, instead of users and calculations are made on the set of users who both rated items i and j (denoted as $U_i \cap U_j$), instead of set of items.

For example, the Adjusted cosine similarity between the items i and j are computed as follows:

$$AdjustedCos(i,j) = \frac{\sum_{u \in U_i \cap U_j} (r_{ui} - \mu_i) \cdot (r_{uj} - \mu_j)}{\sqrt{\sum_{u \in U_i \cap U_j} (r_{ui} - \mu_i)^2} \cdot \sqrt{\sum_{u \in U_i \cap U_j} (r_{uj} - \mu_j)^2}},$$

where μ_i is the average rating of the i-th item.

Once we compute similarity values between items, we can predict the ratings using some of prediction functions. Like in the user-based method, the item-based method also commonly us **weighted sum** and **regression models** to calculate prediction. Weighted sum for item-based approach calculates the prediction of an item i for user u by computing the sum of the ratings that are weighted by the corresponding similarity between the item i and it's k similar items:

$$\hat{r}_{ui} = \frac{\sum_{j \in K_i(u)} Sim(i,j) \cdot r_{uj}}{\sum_{j \in K_i(u)} |Sim(i,j)|}, \tag{1.4}$$

where $K_i(u)$ is the k most similar items to target item i rated by user u. In other words, this approach tries to catch how the target user rates similar items. The weighted sum is divided by the sum of the similarity items to ensure that the prediction is within a predefined range.

The item-based approach is similar to the user-based approach, except that regression uses correlations between items rather than user-user correlations [6]. To model the rating prediction of user u for target item i we can substitute the normalized similarity coefficient from Equation 1.3 with the unknown parameter w_{ij}:

$$\hat{r}_{ui} = \sum_{j \in K_i(u)} w_{ij} \cdot r_{uj},$$

The set $K_i(u)$ is the most similar items to target item i and can be determined using some if the similarity functions, such as Pearson correlation coefficient or adjusted cosine measure.

As in user-based regression models, to calculate unknown parameter w_{ij} we can use the aggregate squared difference between the predicted rating $\hat{r}_{ui}$ and existing r_{uj} to create the least-squares objective function which can be defined as the sum of squared errors of prediction.

1.2.3 Comparing User-Based and Item-Based Methods

Although user-based and item-based methods seem very similar at first glance, they have some important differences:

- *Accuracy:* Item-based methods provide predictions for items similar to target items based on the user's own ratings. For example, similar items to a target horror movie might be a set of other horror movies. In the user-based methods, the ratings are extrapolated from other users who may have different but overlapping interests. As a result, item-based methods often give more relevant recommendations and so have better accuracy.

On the other hand, different interests may lead to greater diversity in the recommendation process for user-based methods. Greater diversity also encourages serendipity, which can reveal unexpected and interesting items for users. Item-based methods can sometimes recommend obvious items or items which are not novel from previous user experiences. Without sufficient novelty, diversity, and serendipity, users might get bored with very similar recommendations they've already looked at [6].

Also it is worth noting that the relative accuracy between these methods can also depend on the datasets.

- *Computational cost:* On most of the e-Commerce systems, such as Amazon or eBay, the number of users is generally much larger than the number of items. In such cases, two users may have a very small number of co-rated items, but two items are more likely to have a larger number of users who have mutually rated them. So adding multiple ratings can't radically change the similarity values between items and the item-item similarity matrix can be calculated offline, that in turn reduce the computational cost of online prediction.

  This does not apply to user-based methods, where adding multiple ratings can radically change the similarity values. Also, based on the facts that new users are more likely to be added than new items, user neighborhoods should be computed more frequently with the addition of new users. As a result, calculating the similarity between users cannot be moved offline [23].

- *Providing explanations:* Item-based methods can provide a concrete reason for the recommendation, that can be explained using the item neighborhoods. On the other hand, these explanations are harder to address with user-based methods, because the peer group is just a set of anonymous users (because of privacy concerns) and it is hard to explain how recommended items relate to user tastes or to those of friends he knows and trusts [6].

1.2.4 Summary

The main advantages of memory-based techniques that they are easy to implement, they are context independent and compared to other methods, such as content-based, it is more accurate. But these systems have some problems:

- *Rating sparsity:* The percentage of people who rate items is generally low. One way to overcome this problem is to use information of user profile when calculating user similarity. That is, users could be considered similar not only if they rated the same item similarly, but also if they belong to the same demographic segment [5]. This extension of traditional collaborative filtering approaches is called “demographic filtering”. For example, [18] uses gender, age, education, and employment information of users in the restaurant recommendation application.

- *Cold-start for new users:* New users have no information about them to be compared with other users. Using a hybrid recommendation approach, that combines collaborative filtering and content-based methods can solve this problem.

- *New item:* Like for new users, no user ratings are available for the new item. The problem can also be addressed using hybrid recommender systems, that are described in section 1.1.4 in more detail.

- *Scalability:* The more users there are in the system, the greater will be the cost of finding the nearest k neighbors.

Some of these problems try to overcome model-based models, that are discussed in the next chapter.

1.3 Model-based collaborative filtering

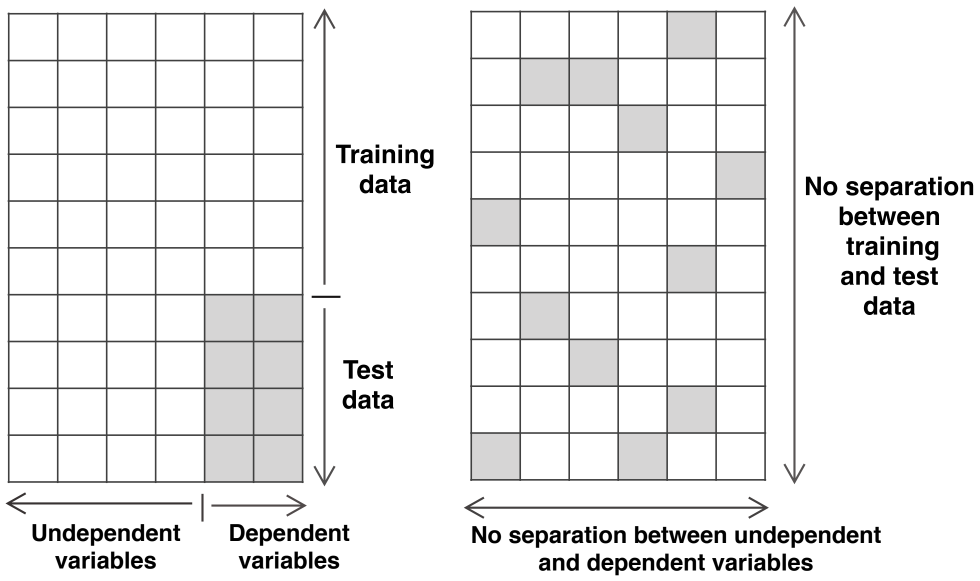
In contrast to memory-based collaborative filtering methods, which utilize the entire user-item matrix to make recommendations for active users, a model-based approach first uses the advantages of data mining and machine learning techniques to learn a predictive model from training data. That model can characterize the rating behavior of target users. Then model-based method uses the trained model to make predictions, rather than directly utilize the entire user-item matrix to compute predictions [4].

Typical examples of model-based filtering methods include decision trees, association rules and latent semantic models. These methods are discussed in detail in following subsections.

1.3.1 Decision trees

Decision trees are frequently used in data classification. In the data classification problem, we have a $m \times n$ matrix, in which the first k columns are feature (independent) variables with fully specified entries, and the last $(n - k)$ columns are the dependent variables, where only a subset of the entries is observed (see Figure 1.1a). These fully defined variables are referred to as the training data, the remaining ones are called test data that need to be examined.

Unlike data classification, where clear separation exists between the training and test data, any entry in the rating matrix may be missing (the shaded entries in Figure 1.1b). Thus, it can be seen that the matrix completion problem is a generalization of the classification problem [6].


Training data
Test data
Undependent variables
Dependent variables
No separation between training and test data
No separation between undependent and dependent variables

(a) Classification problem (b) Collaborative filtering

Figure 1.1: Differences between classification problem with collaborative filtering, shaded entries need to be predicted.

Before explaining decision trees to collaborative filtering, it would be better and easier to describe the application of decision trees to classification problem and assuming that all variables are binary. The decision tree is a hierarchical partitioning of the data space using the split criteria in the independent variables. For example, in a binary matrix R, most of the records relating to the different classes will be separated into two branches – feature variable that takes the value 0 will lie in one branch, others will lie in the other branch. Thus the binary decision tree will be created.

Algorithms for constructing decision trees work top-down by selecting an attribute at each step that best splits the set of items at a given level of the tree. For measuring the ”best” attribute the Gini index is used. The Gini index of node S is defined as follows:

$$G(S) = 1 - \sum_{i=1}^{r} p_i^2,$$

where r is the number of different classes in a node S and p_i is the fraction of data records belonging to class i. Using the weighted average Gini index of the child nodes created from a split we can evaluate the quality of the split:

$$Gini(S \Rightarrow [S_1, S_2]) = \frac{n_1 \cdot G(S_1) + n_2 \cdot G(S_2)}{n_1 + n_2}$$

The final result is a binary tree with decision nodes and leaf nodes, where independent variables are used to map a path from the root to the leaf to classify a test instance with an unknown value [6]. The root node of the tree is the topmost decision node which corresponds to the best predictor.

Then the variables are numerical values, the attribute values can be divided into intervals to perform the splits. In order to comply with numeric dependent variables, the Gini index often changes to other variants of the split criteria, such as error rate, entropy, and variance.

The described approach of decision trees is used in the data classification problem, where exists clear separation between the training and test data. To apply this method to collaborative filtering, where any entry in the rating matrix may be missing, the approach must be modified. While the dependent and independent variables are not differentiated in the collaborative filtering, which item should be predicted by the decision tree?

The solution is to create a lower-dimensional representation of the data using the dimensionality reduction methods, such as *maximum likelihood estimation* (MLE). In this approach, the MLE of the covariance between each pair of items in the rating matrix is estimated as the covariance between only the specified entries [6]. So only the users that have co-rated items are used to estimate the covariance, in other cases, the covariance will be 0. Then the resulting $n \times n$ covariance matrix will be reduced to $n \times d$ basis matrix E_d by selecting the top-d eigenvectors. The next step is projecting the ratings of each user on the eigenvectors by computing the averaged contribution a_{ui} of user u on the ith column of E_d:

$$a_{ui} = \frac{\sum_{j \in I_u} r_{uj} e_{ji}}{|I_u|} \tag{1.5}$$

where e_{ji} is jth entry of ith column of E_d, I_u is the set of item rated by user u, r_{uj} is observed rating from R. The resulting $m \times d$ matrix A is reduced and completely specified matrix.

To predict rating for item j, jth column will be excluded from rating matrix R and from the right-hand side of Equation 1.5. The resulting $m \times t$ representation, in which $t \ll (n-1)$ is used to construct the decision tree for the jth item.

1.3.2 Rule-based collaborative filtering

Association rule mining is a rule-based method that uses machine learning models for discovering relationships between variables in large databases. Association rules are created by searching data for frequent if-then itemsets (patterns) and using the *support* and *confidence* criteria to determine the set of items, that are closely correlated in the database. *Support* is an indication of how frequently the itemset appears in the database; *confidence* indicates the number of times the if-then pattern turned out to be true, in other words it measure the "strength" of a rule.

An association rule is defined as an implication of the form $X \Rightarrow Y$, where X is called antecedent (if), Y is called consequent (then), the "$\Rightarrow$" is the direction of the correlation between X and Y.

The process of finding association rules is a two-phase algorithm, and it requires to specify minimum support and minimum confidence. First, all frequent itemsets in a database that satisfy minimum support are determined. Then a minimum confidence constraint is applied to retain only satisfying rules.

Association rules algorithms can be used as tools for collaborative filtering to build recommender systems and they are particularly useful for performing recommendations in the context of unary rating matrices, that are created by customer activity (e.g., buying items or watching a video) [6]. For example, the items purchased by a customer are set to 1, while the unobserved items are set to 0. This type of explicit feedback is a natural mechanism for expressing user preferences for an object. In this case, setting missing values to 0 is considered an acceptable practice in sparse unary matrices, because of no mechanism to specify a dislike. However, it is not common for most types of rating matrices because rating with zero value will be regarded as a bad rating and it will lead to bias in the predictions.

In the rule-based collaborative filtering, only rules in which the consequent contains exactly one item are retained. Consider an active client who needs to recommend the most relevant products. The first step is to retain rules, where itemset in the antecedent is a subset of the items preferred by target customer (also called "fired" rules by the customer). Then these rules are sorted in order of decreasing confidence, and the first k items discovered in the consequents of these sorted rules are recommended as the top-k items to the target customer [6].

This basic approach can be extended to be able to specify the user's dislikes. When the number of possible ratings is small, each value of the rating-item combination can be treated as a pseudo-item [6]. For example in the POI recommender system such pseudo-items can be (Item = Starbucks, Rating = Like), (Item = KFC, Rating = Dislike). Then the rules are created in terms of these pseudo-items by using the basic approach as discussed above. Such rules may look like the following:

$$(Item = Starbucks, Rating = Like) \Rightarrow (Item = Costa\,cofee, Rating = Like)$$
$$(Item = Starbucks, Rating = Like)\ \&\ (Item = KFC, Rating = Dislike) \Rightarrow$$
$$(Item = McDonald's, Rating = Dislike)$$

For a target customer, the set of ”fired” rules is determined by identifying the rules whose antecedents contain a subset of the pseudo-items for that user. The rules are sorted in order of reducing confidence and then can be used to predict ratings for items by selecting the top-k pseudo-items in the consequents of these rules.

1.3.3 Latent factor models

Latent factor models are considered to be one of the popular approaches in recommender systems. The basic idea of these models is to take advantage of the fact that a significant part of the rows and columns of data matrices are highly correlated and as a consequence, the data has great redundancies. Due to the built-in redundancies, the resulting data matrix can be approximated by a fully specified low-rank matrix, that can be determined even with a small subset of the entries in the initial matrix.

These models use principles of well-known dimensionality reduction methods to fill in the missing values, such as principal component analysis (PCA) and singular value decomposition (SVD). In the geometric interpretation, dimensionality reduction methods typically represent the projection of the data on p-dimensional hyperplane as an approximation (after removing noisy variations), where positively correlated items will mostly be arranged along the same p-dimensional latent vectors that define the hyperplane. Put it shortly, the basic idea of all these methods is to find latent factors, in which the average squared distance between the data points and hyperplane is as small as possible and then reduce the dimensions.

On its pure form, these dimensionality reduction methods would only work with a fully specified matrix, and they are not particularly helpful for the estimation of missing values, because the rating matrix R is very sparse (more than 95 % of the values are missing). The first option that can come to mind is to fill the missing records with some simple heuristics, such as the average value of columns (or rows). Once the matrix is dense, we can use traditional SVD algorithms. The problem of this approach that the results are usually highly biased. Therefore, another approach based on a minimization problem will be used, and before describing it, it is necessary to explain the basic principles of matrix factorization.

In the basic matrix factorization model, the $m \times n$ rating matrix R is approximately factorized into an $m \times k$ matrix U and an $n \times k$ matrix V , as follows [6]:

$$R \approx UV^T \tag{1.6}$$

Each column of U and V is referred to as a *latent vector*, whereas each row of U and V is called *latent factor*. The ith row $\overline{u_i} = (u_{i1} \dots u_{ik})$ of U is referred to as a *user factor* and the jth row $\overline{v_j} = (v_{j1} \dots v_{jk})$ of V is referred to as an *item factor*.

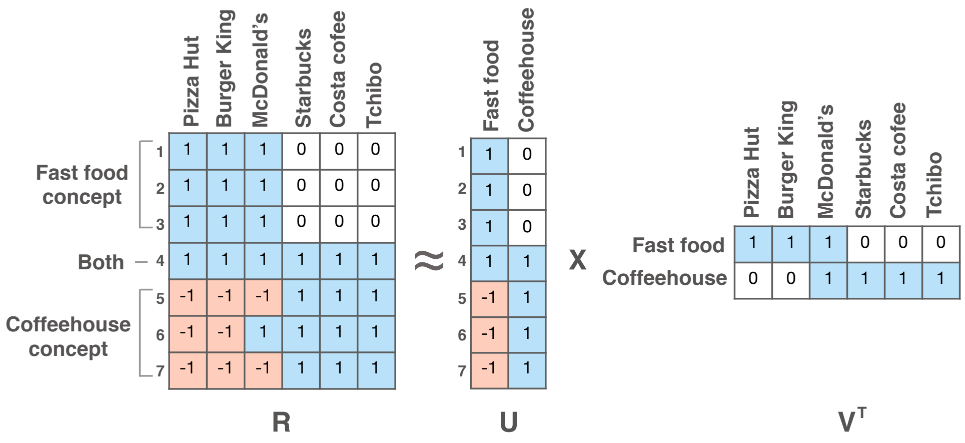


Figure 1.2: Example of matrix factorization into rank-2 factors (concepts)

Let's consider the 7×6 ratings matrix R that illustrated in Figure 1.2. In this matrix, every column corresponds to different item and every row corresponds to different user. In our case the items are restaurants that can be classified into two types – fast food and coffee shops. For simplifying, we will consider only three types of ratings – like, neutrality and dislike (i.e., 1, 0, -1 in numerical representation). Matrix R can be approximately factorized into U and V matrices, as shown in Figure 1.2. After analyzing the matrix R, all users can be divided into two concepts - fast food lovers and coffee shop lovers (only the fourth user belong to both at once). These relations can be seen at matrix U, where each row $\overline{u_i}$ show how much a given user corresponds to a given concept. So in some sense matrix, U can be considered as "user-to-concept" matrix. As for matrix V, each row $\overline{v_j}$ represents the affinity of the jth item towards the concept and can be considered as "item-to-concept" matrix.

From Equation 1.6 follows that each rating r_{ij} in R can be approximately expressed as a dot product of the ith user factor and jth item factor [6]:

$$\hat{r}_{ij} \approx \overline{u_i} \cdot \overline{v_j} = \sum_{s=1}^{k} u_{is} \cdot v_{js}$$

Finding the factor matrices U and V can be done by solving the following optimization problem:

$$\min_{u,v} \frac{1}{2} \sum_{r_{ij} \in R} (r_{ij} - \overline{u_i} \cdot \overline{v_j})^2 = \frac{1}{2} \sum_{r_{ij} \in R} \left(r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js} \right)^2$$

Generally speaking, we want to find vectors $\overline{u_i}$ and $\overline{v_j}$ that makes the sum minimal and trying to match the values r_{ij} of incomplete matrix R as closely as possible. Once we know the values of the vectors $\overline{u_i}$ and $\overline{v_j}$, we can reconstruct U and V and then fill in any missing values for the users rating in matrix R. Note, that the aforementioned objective function is computed only over the observed entries in R.

One of the main problems with this approach that in real settings the rating matrix R is very sparse. In such cases, the relatively few entries are observed, and it can cause overfitting. A common approach to solving this problem is to use regularization. The idea is to discourage very large values of the coefficients in U and V by adding the regularization term to the objective function:

$$\min_{u,v} \frac{1}{2} \sum_{r_{ij} \in R} \left(r_{ij} - \sum_{s=1}^{k} u_{is} \cdot v_{js} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{m} \sum_{s=1}^{k} u_{is}^2 + \frac{\lambda}{2} \sum_{j=1}^{n} \sum_{s=1}^{k} v_{js}^2$$

where $\lambda > 0$ is the regularization parameter that controls the weight of the regularization term. This regularization approach reduces the tendency of the model to overfit by introducing a bias in the model.

The most popular and effective ways to solve these optimization problems is to use the gradient descent or stochastic gradient descent approach [24], or alternating least squares methods [25]. The discussion of these algorithms is beyond the scope of this work, the detailed discussion may be found in [26, 27]

1.3.4 Summary

This section discusses model-based collaborative filtering methods, such as decision trees, rule-based collaborative filtering, and latent factor models. Latent factor models are state-of-the-art in collaborative filtering, and one advantage of using this approach is that instead of having a high dimensional and sparse matrix we will deal with a much smaller matrix in a lower-dimensional space. The use of decision trees for recommendation models also offers several benefits, such as interpretability and flexibility in handling a variety of input data types (ratings, contextual, etc.) [12]. Systems based on association rule-based have found significant applications in web-based recommender systems. The approach is naturally suited to Web personalization systems, as it is specifically designed for sparse transaction data that is commonly encountered in web click behavior [6].

Model-based algorithms tend to be faster than memory-based algorithms because the time required to query the model is usually much smaller than the time required to query the whole dataset. But its disadvantages are that many theoretical models are complex and so not to easy to implement.

1.4 POI Recommendation on Location-Based Social Networks

Location-based social networks (LBSNs) have become very popular and attracted lots of attention from internet users, business and academia with the increasing popularity of GPS-enabled mobile devices. In LBSNs, users can build connections with their friends, upload photos, leave tips and share their locations by checking in to points of interest (e.g., restaurants, famous landmarks, housing locations and parking spaces) [1]. Typical location-based social networks include Foursquare, Yelp, Facebook Place, GeoLife, etc.

POI recommendation helping users filter out uninteresting POIs and reduce their decision-making time. In addition, POI recommender systems can not only help users to find new POIs of their interests but also help to increase revenues of businesses by providing users with intelligent location services, such as location-aware advertisements [1].

Compared with traditional recommender systems, POI recommender systems have the following unique characteristics:

- *Data sparsity:* Unlike traditional recommender systems, where users generally expressed their preferences by explicitly providing ratings for items, a user's preferences are often reflected by the frequency of check-in in locations. The frequency data have a wide range compared with explicit ratings because the number of POIs visited by the individual user is usually only a tiny part of all POIs. For example, the density of the Foursquare dataset used in our experimental studies is around 0.05%, while the density of Netflix dataset for movie recommendations is about 1.2% [28]. This issue leads to negative effects on the recommendation quality of recommender systems based on the collaborative filtering approaches.

- *Geographical influence:* In many cases, users tend to visit nearby locations or POIs within a radius of activity regions, for example near home, workplace or university. Therefore the probability of visiting new places decreases as the distance increases. So geographical influence is the essential characteristic that distinguishes POI recommender systems from traditional recommender systems and profoundly effects users' visiting behaviors [1].

- *Temporal influence:* Users' preferences are time-dependent – users visit different places at a different time in a day. For example, in the early morning, people will more often go to coffee than to a restaurant. Besides, users' check-in behavior can vary depending on opening hours and peak hours of different POIs (bars vs. coffee shop). So temporal influence can also affect users' visiting preferences.

- *Social influence:* Based on the assumption that users' visiting preferences might be affected by their social ties, several studies [29, 30] tried to combine social relationships with ratings and shown that it can improve the quality of the recommendation. However, previous research [31] has presented that about 96% of users share less than 10% common visited interests [1]. Hence, social influence has a limited effect on users' check-in behaviors in terms of POI recommendation.

1.4.1 Notations

In the rest of this work will be used the following notations. Typical POI recommender system consists of a set of M users $U = \{u_1, u_2, ..., u_M\}$, and a set of N locations (i.e., POIs) $L = \{l_l, l_2, ..., l_N\}$. Each user is associated with a set of POIs L_u visited by the user. Users and locations have unique identifiers, also each location has GPS coordinates – longitude and latitude. Information of users' check-in is converted to user-location check-in frequency matrix C. Each entry c_{ul} of C represents the number of check-ins made by user u for location l. In practical, frequency matrix C is very sparse because the number of POIs visited by the individual user is usually only a tiny part of all POIs. In cases when check-in information is not available, users' ratings are used and converted to a user-location rating matrix R, with M rows and N columns. Each entry r_{ul} of R represents the rating given by user u for location l. Generally, ratings are integers and fall into the closed interval, e.g., from 0 to 5, where a higher rating corresponds to a more positive attitude to a POI. Like matrix C, rating matrix R is usually very sparse. Also, information about social relationships of users is transformed into matrix S, where $s_{uv} = 1$ means the existence of the social relationship between user u and v, and zero value is vice versa. In this work, $|| \cdot ||$ will denote the (squared) Frobenius norm of the matrix.

Next section in details will review existing researches and algorithms for POIs recommendation.

1.5 Related works

With the availability of the vast amount of users' visiting history, the problem of POI recommendations has been widely studied and received significant research attention in the last seven years, and many approaches have been suggested. These studies vary in types of additional context information (social, geographical or temporal), problem settings, recommendation models and so on.

According to the type of additional context information combined with check-in data, POI recommendation algorithms were classified into four categories: geographical influence, social influence, and temporal influence enhanced POI recommendation approaches.

Geographical enhanced POI recommendation approaches assume that users tend to visit nearby locations or POIs within a radius of activity regions, for example near home, workplace or university. Temporary influence implies that users' preferences are time-dependent and it can affect users' visiting preferences. Social influence enhanced approaches are based on the assumption that users' visiting preferences might be affected by their social ties.

The rest of this section will review the existing research works classified to each category.

1.5.1 Geographical influence

Geographical influence is the essential characteristic that distinguishes POI recommender systems from traditional recommender system. Based on the assumption that users tend to visit nearby locations within a radius of activity regions, a lot of studies tried to leverage geographical influence in users' check-in activities and they shown that it profoundly effects users' visiting behaviors.

In the work [32], Ye at al. studied the implication of distance on user check-in behavior by estimating the probability density function over the distance between all pairs of POIs where the user has checked in. They employed a power-law distribution to model users check-in behaviors and proposed a collaborative POI recommendation algorithm based on the naive Bayesian method.

On the other hand, Yuan et al. [33] made a different assumption – the willingness that a user moves from one POI to another one is a function of their distance. The willingness of the user to visit POI at a distance of d km is defined as follows (a and k are parameters of the power-law function):

$$w(d) = a \times d^k$$

Consider a user is currently at POI l_i and POI l_j is a candidate to check in at distance $dis(l_i, l_j)$ from l_i. The probability that the user will check in l_j is computed using the following equation:

$$p(l_j|l_i) = \frac{wi(dis(l_i, l_j))}{\sum_{l_k \in L, l_k \neq l_i} wi(dis(l_i, l_j))}$$

Note that as the distance increases, the conditional probability decreases, which reflects that the user will visit a distant POI in the lower probability.

Their experiments were shown that the proposed POI recommendation works much better than [32] in terms of precision and recall.

Liu et al. [34] proposed a geographical probabilistic factor analysis framework for POI recommendation by combining geographical influence with Bayesian non-negative matrix factorization (BNMF) [1]. They used a Gaussian distribution to represent a POI over a sampled region, reflecting the first law of geography, that says: "Everything is related to everything else, but near things are more related than distant things" (Tobler 1970). Their experimental results show that the proposed method outperforms approaches based on regularized [35] and probabilistic [36] matrix factorization.

Recently, Lian et al. [37] proposed a POI recommendation approach based on weighted matrix factorization, named GeoMF. GeoMF divides the whole geographical space into R grids, each representing a geographical region. Then method expands users' latent feature vectors with activity area vectors of users (matrix X) and also expands POIs' latent feature vectors with influence area vectors of POIs (matrix Y). In detail, the entry y_{jr} of Y that defined as:

$$y_{jr} = \frac{1}{\sigma} K(\frac{dis(r,j)}{\sigma}),$$

represents influence of POI j on region r, where $dis(r,j)$ is the distance between POI j and region r, σ is the standard deviation and $K(\cdot)$ denotes standard normal distribution. Each entry x_{ir} of X represents the probability of the user i appearing in the region r. The objective function of GeoMF is defined as follows:

$$\min_{U,V,X\geq 0} ||W \odot (R - UV^T - XY^T)||^2 + \alpha(||U||^2 + \beta||V||^2) + \lambda||X||_1$$

W denotes the weighted matrix where each entry w_{ui} represents the confidence of user u for POI i. Matrices U and V are users' latent vectors and POIs' latent vectors, respectively. Geographical preference of user i on POI j is estimated by $x_i \cdot y_j$, and the final recommendation score can be approximately expressed as $\hat{r}_{ij} \approx \overline{u_i} \cdot \overline{v_j} + x_i \cdot y_j$.

Their experiments show that the matrix factorization model based on 0/1 rating matrix give better results than the same model based on the frequency matrix. Also, their proposed weighted matrix factorization is superior to other types of matrix factorization models discussed in this work, such as [34, 35, 36].

RankGeoFM [38] is a ranking-based matrix factorization model that studies the preferences of users and includes the geographical influence of neighboring POIs. To represent user preferences as traditional matrix factorization methods do, check-in matrix C is factorized onto two matrices $U^{(1)}$ and V. In addition, RankGeoFM introduces one extra latent factor matrix $U^{(2)}$ to model the interaction between users and POIs for incorporating the geographical influence. Given user i and POI j, the recommendation score is computed as follows:

$$\hat{r}_{ij} = u_i^{(1)} v_j^\intercal + u_i^{(2)} \cdot \sum_{k \in \mathcal{N}(j)} w_{jk} v_k^\intercal$$

On this equation, the first term models the user preference score, while the second term models the geographical influence score that a user likes a POI because of its neighbors [38]. The W denotes $N \times N$ matrix of the geographical influence, where w_{ij} is the probability of visiting POI l_i, given that POI l_j has been visited. Each entry w_{ij} is set to $(0.5 + dis(l_i, l_j))^{-1}$ if $l_j \in \mathcal{N}(l_i)$, and 0 otherwise. $\mathcal{N}(l_i)$ is the k-nearest neighbors of each POI l_i. Their experimental result shows that RankGeoFM performs a better result than GeoMF [37] and achieve approximate 14 % improvement.

1.5.2 Temporal influence

Traditional recommendation systems use temporal influence as a factor that turns the weights of ratings. In contrast, POI recommendation systems typically use it to make POI recommendation for a specific temporal state.

The work by Yuan et al. [33] proposes a time-aware POI recommendation algorithm that extends the user-based collaborative filtering method by exploiting other user's temporal preferences. The basic idea is to split time into hourly-based slots and model the user temporal preference to POIs in a certain time slot.

In this work the temporal behavior similarity between two users is computed with cosine similarity using binary check-in matrix, i.e., if the user has checked in POI at time t the value will be 1, otherwise 0. Then the recommendation score that the user check in a new POI at time t will be computed using weighted sum method. In their experiments on Foursquare and Gowalla datasets was shown that the approach always outperforms the user-based POI recommendation approach, which ignores the time information.

Other work [35] proposes the time-enhanced matrix factorization model that represents each user by different latent vectors for different time slots, and the final recommendation score is computed from all the latent vectors. To represent user preferences at a different time, check-in matrix C is factorized for each time slot t separately onto two matrices U_t and V, where $t \in \{0, 1, \ldots, 23\}$ is an hour of the day. U_t reprezent a time-dependent user check-in preferences under temporal state t, and V represent a time-independent characteristics of POIs. Finding these matrices can be done by solving the following optimization problem:

$$\min_{U_t \geq 0, V_t \geq 0} \sum_{t=1}^{T} ||Y_t \odot (C_t - U_t V^T)||^2 + \alpha \sum_{t=1}^{T} ||U_t||^2 + \beta ||V||^2$$

Furthermore, they modeled the temporal consecutiveness property by added a temporal regularization term into the objective function of matrix factorization:

$$\sum_{t=1}^{T}\sum_{i=1}^{N}\psi_i(t,t-1)||u_i^{(t)}-u_i^{(t-1)}||_2^2,$$

there $\psi_i(t,t-1)$ is the temporal coefficient that measures the similarity of user preferences between temporal state t and $t-1$, i.e. between C_i^t and C_i^{t-1}, $u_i^{(t)}$ is the ith row of matrix U^t.

1.5.3 Social influence

Based on the assumption that users' visiting preferences might be affected by their social ties, several studies tried to combine social relationships with check-in information and shown that it can improve the quality of the recommendation.

In the work [31], Ye et al. develop a friend-based collaborative filtering (FCF) approach for POI recommendation system based on collaborative check-ins made by social friends. First, the FCF finds k most similar friends among all friends, so it takes into account only the preferences of friends, instead of every user. To predict the rating, the FCF uses a weighted sum method. Their experiments were shown, that FCF brings minor improvement over user-based POI recommendation.

In their later work [32], Ye et al. deduced the social influence weight between friends based on social connections and similarity of their check-in behavior:

$$w_{i,k}=\eta\frac{|F_k\cap F_i|}{|F_k\cup F_i|}+(1-\eta)\frac{|L_k\cap L_i|}{|L_k\cup L_i|},$$

where η denotes a tuning parameter and F_k is the friends of user u_k.

The work [36] includes the social information into the probalistic matrix factorization (PMF) to improve the model performance. The objective function is defined as follows:

$$\min_{U,V}\sum_{i=1}^{N}\sum_{j=1}^{M}I_{ij}(g(c_{ij})-g(u_i\cdot v_j))^2+\lambda_1||U||^2+\lambda_2||V||^2+\beta\sum_{i=1}^{N}\sum_{f\in F_i}sim(i,f)||u_i-u_f||^2$$

where $g(x)=1/(1+e^{-x})$ is the logistic function, $sim(i,f)$ is the similarity between user u_i and his friend u_f, I_{ij} is the indicator function which equals to 1 if user i checks in the location j and 0 otherwise.

An interesting approach was suggested in [39] by Wanga et al. They propose a link-based model that constructs a graph to represent user preference (check-in behaviors) and social influence (friendship relations) by different types of edges. Based on the constructed graph, the Bookmark-Coloring Algorithm algorithm is executed to calculate the similarity between users and then the user-based collaborative filtering is executed.

The experimental results of the above-mentioned approaches show that social influence has less weight than geographical influence and check-in activities because most friends have small overlapping on their check-in POIs.

Chapter 2

Content-aware POI recommendation algorithm

While previous major works of POI recommendation on LBSNs mostly focuses on investigating the spatial, social and temporal patterns of check-in behavior of users, the use of additional content information available in LBSNs has not been systematically studied. The various types of content information can be associated with different aspects of a user's check-in behavior, giving a unique opportunity for POI recommendation. In addition, the use of content information can deal with cold start item issue in recommendation systems, which is especially relevant in POI recommendation.

In this work, we study the impact of content information available in LBSNs on POI recommendation. As a baseline approach, we use matrix factorization methods, due to their good scalability and predictive accuracy. Also, the matrix factorization technique offers a flexible structure to incorporate additional sources of information to improve recommendation quality [4]. In our case, we will use content information, such as POI categories and attribute information and construct content-aware POI recommendation algorithm.

The following section is organized as follows. First, we introduce the basis of the content-aware POI recommendation algorithm. The model will be explained using traditional explicit feedback – rating matrix. Because of different LSBNs provide datasets with a different type of feedback (for example Foursquare provide user check-in history, i.e., implicit feedback; Yelp provide user ratings, i.e., explicit feedback), in section 2.2 we clarify differences between implicit and explicit feedback and explain the need for using different approaches to model user preferences. Finally, we present the content-aware POI recommendation algorithm for implicit feedback.

2.1 Algorithm description

As a baseline approach, we use the state-of-the-art matrix factorization method (described in the section 1.3.3), that offers a flexible structure to incorporate additional sources of information to improve recommendation quality. Also, by comparison with other popular collaborative filtering approaches, matrix factorization shows the best scalability and predictive accuracy.

Recall that the regularized objective function which works with incomplete matrices, is computed only over the observed entries in rating matrix R as follows:

$$\text{Minimize } J = \min_{U,V} \frac{1}{2}||R - UV^T||^2 + \frac{\lambda}{2}||U||^2 + \frac{\lambda}{2}||V||^2 \tag{2.1}$$

Here, $m \times k$ matrix U and $n \times k$ matrix V are the unknown matrices, which need to be learned to minimize the objective function. The most popular and effective ways to solve this biconvex optimization problem is to use the gradient descent or stochastic gradient descent approach (SGD) [24] that applied to seek a local minimum solution of the objective function. To learn the user latent feature matrix U one needs to compute the partial derivative of J with respect to the decision variable u_i with fixed V:

$$\frac{\partial J}{\partial u_i} = (R_{ij} - u_i \cdot v_j)(-v_j) + \lambda u_i \tag{2.2}$$

And similarly for matrix V we fix U and compute the partial derivative of J with respect to the decision variable v_j:

$$\frac{\partial J}{\partial v_j} = (R_{ij} - u_i \cdot v_j)(-u_i) + \lambda v_j \tag{2.3}$$

Subsequently, the updates can be computed as follows:

$$u_i \leftarrow u_i + \alpha((R_{ij} - u_i \cdot v_j)(v_j) - \lambda u_i)$$
$$v_j \leftarrow v_j + \alpha((R_{ij} - u_i \cdot v_j)(u_i) - \lambda v_j)$$

where $\alpha > 0$ is the step size, which often is set to small constant value or can be chosen using standard numerical methods in nonlinear programming.

The SGD algorithm continues iterating on the training set multiple times until it reaches convergence or exceeds the upper limit of the number of iterations. Note that it is necessary to update all entries in both matrices simultaneously with the use of auxiliary variables to store intermediate results during an update.

The next step is to utilize content information to regularize the matrix factorization. In the work [4], Yu, Wang, and Gao propose to add the following item relationship regularization term based on item attribute information to constrain the baseline matrix factorization framework:

$$\frac{\beta}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} S_{ij}||v_i - v_j||^2$$

where β is regularization parameter to control the impact from the item attribute information, S is similarity matrix where $S(i, j)$ represents the similarity between items i and j based on their item attribute information.

The key idea of this regularization term is to make two item latent feature vectors more "close" if they share common characteristics based on their item attribute information. A small value of $S(i, j)$ means that the distance of two item latent feature vectors must be great, while a small value of distance indicates that $S(i, j)$ must be large [4]. Using this approach, we can better characterize the item latent feature vectors in the process of matrix factorization, thereby produce more accurate recommendations compared to traditional approaches.

By adding the item relationship regularization term into Equation 2.1, the objective function J changes to:

$$J^* = \min_{U,V} \frac{1}{2}||R - UV^T||^2 + \frac{\lambda}{2}||U||^2 + \frac{\lambda}{2}||V||^2 + \frac{\beta}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} S_{ij}||v_i - v_j||^2$$

The gradient with respect to u_i (Equation 2.2) and corresponding update rule remain the same, while the gradient with respect to v_j in Equation 2.3 changes to:

$$\frac{\partial J^*}{\partial v_j} = (R_{ij} - u_i \cdot v_j)(-u_i) + \lambda v_j + \beta(v_j \sum_{k=1}^{N} S_{jk} - \sum_{k=1}^{N} S_{jk} v_k)$$

Accordingly updating rule for v_j changes to:

$$v_j \leftarrow v_j + \alpha((R_{ij} - u_i \cdot v_j)(u_i) - \lambda v_j - \beta(v_j \sum_{k=1}^{N} S_{jk} - \sum_{k=1}^{N} S_{jk} v_k))$$

To construct similarity matrix S we must compute the similarity between POIs using some of the content information available in LSBNs. In our dataset each venue is represented by a collection of attributes (i.e., WiFi - true/false, price range, parking - true/false, noise level and so on) and categories (i.e., bakery, restaurant, coffee shop), so we will use attributes A and categories C to compute similarity between each pair of POIs. To compute similarity matrix S we construct the following similarity measure:

$$Sim(i, j) = \frac{\sum_{k=1}^{D} \delta(a_{i,k}, a_{j,k})}{D} + \frac{|C_i \cap C_j|}{|C_i \cup C_j|}$$

where D is the number of attributes and $\delta(a_{i,k}, a_{j,k})$ returns 1 if attributes $a_{i,k} = a_{j,k}$ and 0 otherwise.

To summarize, the pseudo-code for the SGD content-aware matrix factorization approach for POI recommendation system is described in the following Algorithm 1.

Algorithm 1: Content-aware matrix factorization algorithm for POI Recommendation System based on explicit feedback

```
Input :
        $\boldsymbol{R}$ : the user-POI rating matrix.
        $\boldsymbol{S}$ : POI similarity matrix computed using Equation 2.1.
        $\boldsymbol{k}$ : the dimension of latent feature vector.
        $\boldsymbol{w}$ : the number of iterations.
        $\boldsymbol{\lambda}$ : the regularization parameter for user regularization term.
        $\boldsymbol{\beta}$ : the regularization parameter for item relationship
        regularization term.
        $\boldsymbol{\alpha}$ : the step size (learning rate).
Output:
        $\boldsymbol{U}$: the user latent feature matrix.
        $\boldsymbol{V}$: the POI latent feature matrix.
Randomly initialize matrices $U$ and $V$;
$R^* = \{(i, j) : r_{ij}$ is observed $\}$;
Initialize $c = 0$;
while not(convergence) or $c < w$ do
    for each $(i, j) \in R^*$ in shuffled order do
        $e_{ij} \Leftarrow r_{ij} - \sum_{s=1}^{k} u_{is} v_{js}$;
        $u_i^+ = u_i + \alpha e_{ij} v_j - \lambda u_i$;
        $v_j^+ = v_j + \alpha e_{ij} u_i - \lambda v_j - \beta(v_j \sum_{k=1}^{N} S_{jk} - \sum_{k=1}^{N} S_{jk} v_k)$;
        $u_i = u_i^+$;
        $v_j = v_j^+$;
    end
    c++ ;
end
return U and V
```

The above algorithm continues iterating on the training set multiple times until it reaches convergence or exceeds the upper limit of the number of iterations. The convergence condition can be computed using the following check $(J^{*[i]} - J^{*[i+1]}) < \varepsilon$, where ε is the precision value.

After computing the matrices U and V, we can predict for target user i the rating for POI j as $\hat{r}_{ij} = u_i \cdot v_j$, where $\hat{r}_{ij}$ symbolizes the predicted rating.

2.2 Matrix factorization for implicit feedback

The previously discussed matrix factorization approach is designed for rating predictions in a typical scenario of recommendation system. However, explicit feedback is not always available. Recommendation systems can derive user preferences from more abundant implicit feedback, which indirectly reflect an opinion by observing user behavior, e.g, purchase or browsing history, search patterns, or even mouse movements. In POI recommendation systems implicit feedback usually expressed in the form of check-ins, a higher visit frequency corresponds to larger confidence of preference for the location. These type of feedback is more natural for LSBN systems.

It is essential to identify the unique characteristics of implicit feedback that prevent the direct use of algorithms developed with explicit feedback. The main characteristics are:

1. No negative feedback. Observing the implicit behavior of users, we can infer which items they probably like, but it is difficult to infer what they didn't like reliably. In terms of POI, check-in datasets just include the locations where users have been and therefore they likely prefer. In other words, it just provides positive examples. This fundamental asymmetry is absent in explicit feedback when users tell us what they like and what they don't. It has several implications. For example, explicit recommenders tend to focus only on the user-item pairs that we know their ratings. Thus, the remaining data are treated as "missing data" and excluded from the analysis. This is impossible with implicit feedback, as focusing only on the gathered feedback will leave us with the positive feedback, greatly distorting the full user profile [40]. Hence, it is also crucial to address the missing data, where the most negative feedback is expected to be found.

2. The numeric value of explicit feedback indicates a preference, while the numeric value of implicit feedback indicates confidence [40]. Systems based on explicit feedback allow the user to express their level of preference, e.g., a star rating between 1 and 5. On the other hand, numeric values of implicit feedback describe the frequency of actions, such as how often a user is visiting a certain restaurant. The numerical value of feedback is useful because it tells us about the confidence we have in a particular observation, i.e., the visiting patterns of higher frequency indicate the preferences of higher confidence. A one-time event can be caused by various reasons that have nothing to do with the user's preferences, while a recurring event is more likely to reflect user opinion.

3. Implicit feedback is inherently noisy [40]. As long as we passively track user behavior, we can only guess at their preferences and true motives. For example, we may consider visiting behavior for an user, but this

does not necessarily indicate a positive view of the venue, i.e., a user that did not visit a certain POI might have done so because she dislikes the venue or just because she did not know about it or was not available to visit it.

Due to the unique characteristics of implicit feedback, it is necessary to design other algorithms for explicit feedback. One of the solutions is to randomly select some negative examples for each user and assign them less weight than the positive ones because the confidence in their negative attitude is less than the positive attitude of the positive examples. In the work [40], Yifan Hu et al. propose to consider all non-visited locations as negative examples when the weights to all negative examples are assigned the same value, i.e., 1. Thus, the weight matrix W can be defined as follows:

$$w_{ui} = \begin{cases} \mu(c_{ui}) + 1, & \text{if } c_{ui} > 0 \\ 1, & \text{otherwise} \end{cases} \tag{2.4}$$

where $\mu(c_{ui}) > 0$ is a monotonically increasing function. Based on this weighted matrix W, the objective function for the implicit feedback is represented as follows:

$$\min_{U,V} \frac{1}{2}||W \odot (C^* - UV^T)||^2 + \frac{\lambda}{2}||U||^2 + \frac{\lambda}{2}||V||^2 + \frac{\beta}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} S_{ij}||v_i - v_j||^2 \tag{2.5}$$

where $\odot$ is the Hadamard product operator, i.e., elementwise multiplication of matrices, C^* is 0/1 matrix, where each entry $c^*_{ui} \in \{0, 1\}$ indicates whether a user u has visited a POI i.

This optimization problem can also be solved using the stochastic gradient descent approach. However, due to the weight setting, the approximation error is summed over all entries in the user-POI matrix, and nonzero entries are weighted more heavily. Therefore, the cost function will contain $M \times N$ terms, where M is the number of users and N is the number of POIs. For typical datasets, $M \times N$ can easily reach several billion. In such cases, stochastic gradient descent algorithm becomes too expensive. Fortunately, the approximate error can be efficiently reduced via Alternative Least Squares algorithm (ALS) and its time complexity for each iteration is in proportion to the total number of visited locations, i.e., the number of non-zero entries in the frequency matrix [37].

The basic idea of the alternative least square approach to use the following iterative approach, starting with an initial set of matrices U and V:

1. Keeping V fixed, we recompute and update all user factors, i.e., solve for each of the m rows of U by minimizing the objective function in the Equation 2.5. By differentiating the objective function we obtain a

regularized linear least squares solution for the individual user factor u_i [40]:

$$u_i = (V^T W^i V + \gamma I)^{-1} V^T W^i c_i^*$$

where W^i is an $N \times N$ diagonal matrix, subject to $W^i_{jj} = w_{ij}$ and c_i^* is a binary rating vector of the user i (ith row of the matrix C^*).

2. Keeping U fixed, solve V by minimizing the objective function similarly. By differentiating the objective function given in equation 2.5 we obtain a regularized linear least squares solution for the individual item factor v_j:

$$v_j = (U^T W^j U + (\gamma + \beta(\sum_{k=1}^{N} S_{jk} - 1))I)^{-1} (U^T W^j c_j^* + \beta(\sum_{k=1\&k\neq j}^{N} S_{jk} v_k^{\intercal}))$$

where W^j is an $M \times M$ diagonal matrix, subject to $W^j_{ii} = w_{ij}$ and c_j^* is a binary rating vector of the POI j (jth column of the matrix C^*). A total of n such least-squares problems need to be executed, and each least-squares problem has k variables (k is the dimension of latent feature vector).

3. Repeat Steps 2 and 3 until a stopping criterion is satisfied. Usually, algorithm continues iterating on the training set multiple times until it exceeds the upper limit of the number of iterations or it reaches convergence, i.e if the difference between the observed root mean squared error on the training dataset is less than some precision value ε. When the iteration stops, the obtained U and M are used to make final predictions on the test dataset.

A computational bottleneck on these update expressions is computing $V^T W^i V$ and $U^T W^j U$, whose naive calculation will require time $O(k^2 n)$ (for each of the m user) and $O(k^2 m)$ for each of the n POI respectively (k is the factor of matrices U and V). In the work [40], Yifan Hu et al. propose a trick to speed up its calculation. In particular, by using the fact that

$$V^T W^i V = V^T (W^i - I) V + V^T V$$

$$U^T W^j U = U^T (W^j - I) U + U^T U$$

the second part $V^T V$ independent of user i and $U^T U$ is independent of POI j, so that it can be precomputed. As for first part, it only requires $O(n_i k^2)$, where n_i is the number of visited locations of user i.

Applying the similar trick to calculate the remaining part $V^T W^i c_i^*$ in update expression for u_i, it gets cost $O(n_i k)$. Consequently, recomputation of u_i is performed in time $O(k^2 n_i + k^3)$, where $O(k^3)$ is the cost of the matrix inversion. The same technique will be applied for individual item factor v_j.

To summarize, the pseudo-code for the ALS content-aware matrix factorization approach for implicit feedback is described in the following algorithm.

Algorithm 2: Content-aware matrix factorization algorithm for POI Recommendation System based on implicit feedback

```
Input :
        $\boldsymbol{C}$ : the user-POI check-in frequency matrix.
        $\boldsymbol{S}$ : POI similarity matrix computed using Equation 2.1.
        $\boldsymbol{k}$ : the dimension of latent feature vector.
        $\boldsymbol{w}$ : the number of iterations.
        $\boldsymbol{\lambda}$ : the regularization parameter for user regularization term.
        $\boldsymbol{\beta}$ : the regularization parameter for item relationship
        regularization term.
Output:
        $\boldsymbol{U}$: the user latent feature matrix.
        $\boldsymbol{V}$: the POI latent feature matrix.
Randomly initialize matrices $U$ and $V$;
Initialize weigh matrix $W$, where $w_{ij} = \mu(c_{ij}) + 1$;
Initialize matrix $C^*$ as 0/1 matrix;
Initialize $c = 0$;
for each $i \in M$ do
    foreach $j \in N$ do $W^i_{jj} = w_{ij}$;
end
for each $j \in N$ do
    foreach $i \in M$ do $W^j_{ii} = w_{ij}$;
end
while not(convergence) or $c < w$ do
    Updating U...;
    $VV = V^T \cdot V$;
    for each $i \in M$ do
        $u_i = (V^T(W^i - I)V + VV + \gamma I)^{-1} V^T W^i c^*_i$
    end
    Updating V...;
    $UU = U^T \cdot U$;
    for each $j \in N$ do
        $v_j = (U^T(W^j - I)U + UU + \gamma I + \beta(\sum_{k=1}^{N} S_{jk} -$
        $S_{jj})I)^{-1}(U^T W^j c^*_j + \beta(\sum_{k=1\&k\neq j}^{N} S_{jk} v_k))$
    end
    c++ ;
end
return U and V
```

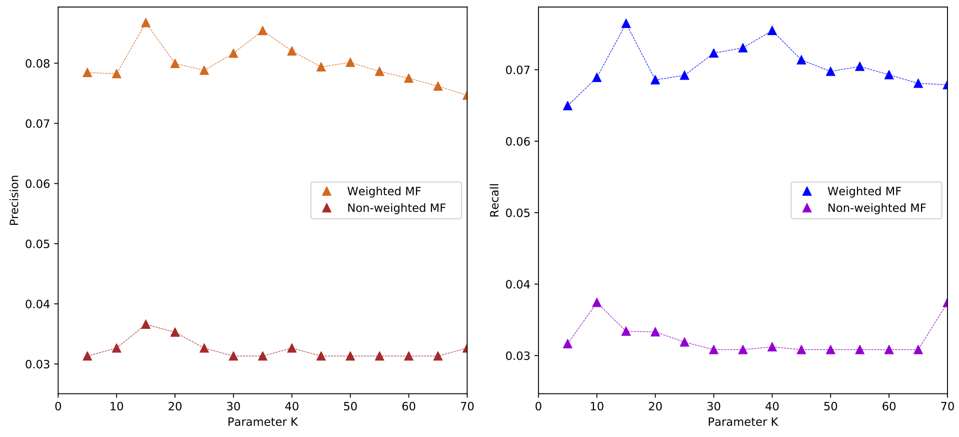
After computing the matrices U and V, we recommend to target user i

the top K POIs with the largest value of $\hat{c}_{ij} = u_i \cdot v_j$, where $\hat{c}_{ij}$ symbolizes the predicted preference of user i for item j. Notice, that recomputation of the user-factors u_i can be done in a parallel fashion since there is no dependence between users. The recalculation of the user-factors is followed by a recomputation of all item-factors v_j and also possible in a parallel way. After the experiments, a maximum number of iterations were set to 10 (see the discussion in Section 4.3). The choice of function $\mu(c_{ij})$ is discussed in Section 4.3.3.

2.2.1 Weighted vs. non-weighted matrix factorization

The main idea of using a weighted method for implicit feedback arose due to the very lower values of precision and recall, that give non-weighted matrix factorization approach. In Figure 2.1 you can see, that prediction accuracy of the non-weighted approach on check-in dataset is very low – the best precision values are around 0.036 and the best recall values are around 0.038 (the baseline regularized matrix factorization defined in Equation 2.1 without adding item attribute information was evaluated).

By adding the weighted matrix proposed by [40] and defined in Equation 2.4 to the objective function, the prediction accuracy is doubled, as you can see in Figure 2.1, the best precision values are around 0.088 and the best recall values are around 0.078.



(a) Impact on precision (b) Impact on recall

Figure 2.1: Prediction accuracy of weighted and non-weighted matrix factorization approach based on implicit feedback

Therefore, the weighted matrix factorization approach for data sets with explicit feedback provides more accurate results that leaves no doubt as to the correctness of the choice of this approach.

Chapter 3

Datasets

Before going into the experiments and evaluation of the algorithm, it is necessary to discuss the existing LBSN datasets. At present, the most popular location-based social networks are Foursquare, Yelp, TripAdvisor, Facebook Place, Mapstr and so on. Unfortunately, not all of them publish their datasets, perhaps due to the privacy concern. Below we discuss the public datasets used in our experiments and the ways of completing missing information required by our algorithm.

3.1 Foursquare

Foursquare[1] one of the most popular location-based social networks that has more than 50 million monthly active users as of October 2018 [2] and keeps growing every month. In this work we use public real-world dataset [2] contains check-ins in New York City collected for about ten months (from 12 April 2012 to 16 February 2013). It contains 227,428 check-ins for 38,333 businesses in New York City. Each check-in is associated with its timestamp, GPS coordinates, and categories of venue. Unfortunately, some venues do not collect a sufficient number of check-ins, and some customers lack adequate check-ins to infer their preferences. Therefore we filter out POIs and customers whose check-ins are less than 10 and exclude useless POIs with category ”Home (private)”, ”Office”, ”Bus Station”, ”Road” and so on. After preprocessing the sparsity of user-POI check-in matrix is 99.231% (before it was 99.872%.).

Foursquare specializes in user check-ins and, thus, social and temporal aspects. But the fousquare dataset has a drawback – it does not contain attribute information about POIs that is required by our algorithm. To avoid such problem, we added missing information to Foursquare dataset using the Yelp API. The process of conflating is described in Section 3.3.

[1] `https://foursquare.com`

[2] Available at `https://sites.google.com/site/yangdingqi/home/foursquare-dataset`

3.2 Yelp

Yelp [3] is a local-search service that focuses on detailed user reviews and a wide range of semi-structured place attributes such as the prices, noise level, wifi availability, etc. The Yelp dataset[4] was publicly released for academic research purposes. It contains interactions between customers and businesses in 10 metropolitan areas, 6.7M reviews for 192,609 businesses. Unlike Foursquare, the Yelp dataset doesn't contain check-ins of users, only the anonymous set of check-ins is available for each venue. Therefore instead of check-in matrix, the rating matrix will be constructed.

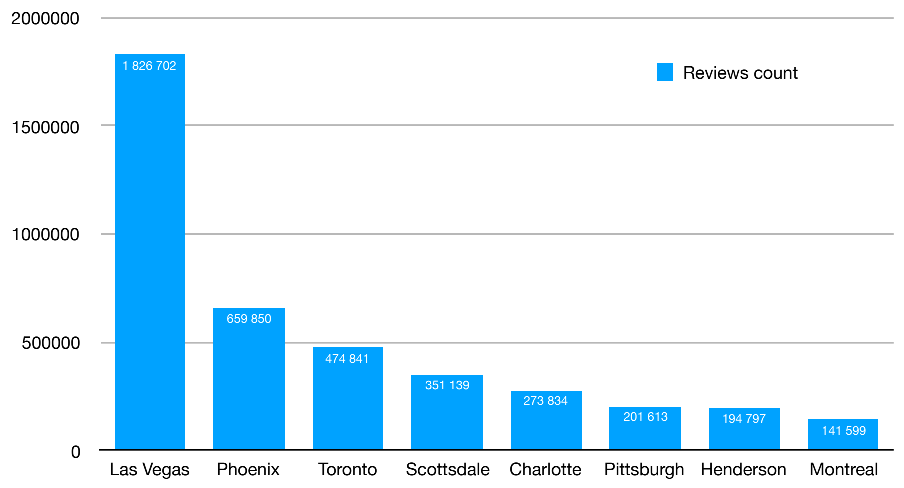


Figure 3.1: Statistics of the number of reviews depending on the city

After summarizing the rating records, we grouped all the rating records (see Figure 3.1) by city and calculated sparsity of each rating matrix. For our experiments, we limited ourselves to businesses and reviews from 4 cities which have the most reviews and simultaneously the densest user-POI rating matrix. After, we filtered out the POIs and customers which have less than 7 ratings. The following table shows the statistics of these cities.

	Las Vegas	**Phoenix**	**Toronto**	**Scottsdale**
# Users	579327	237700	103273	145224
# POIs	28910	18633	18237	8822
Sparsity, %	99,981	99,967	99,974	99,972
After cleaning	99,529	99,397	99.737	99.493

[3] https://www.yelp.com

[4] Yelp dataset challenge Round 13 (access date: March 2019), https://www.yelp.com/dataset/challenge

3.3 Merging datasets

The problem of lack of some information in datasets is very common in recommender-based researches. Each of abovementioned services specializes in certain kinds of place-related information, and both have certain strengths and weaknesses. Therefore, to fully exploit all the variety of information, conflation is required.

From a research perspective, the combination of data sources is desirable for multiple reasons. First, it increases data quality by enriching the data with heterogeneous multi-thematic information. Also, the typos in place names or inaccuracies in GPS coordinates may be corrected. Second, we can exploit additional attributes from multiple data sources to arrive at a more holistic understanding of places. For instance, one can combine user reviews from different communities to study sentiment, compare the place hierarchies and match them using ontology alignment techniques, mine check-in behavior and so on [41].

The process of conflating POIs from data sources can be generally divided into two steps:

1. Establish identity relationships between information entities in different data sources, i.e., whether both entities correspond to the same place in the physical world. To do this, attribute values that are common to both datasets are typically compared using a certain similarity measure. For example, for comparing an attribute with a string value (e.g., name, city), the Levenshtein distance can be used. Other attributes, such as geographic locations, can be compared using appropriate measures. In practice, these measures rarely return exact matches and must be combined using the methods described in the next step.

2. In the second step involves conflating the places using the weighted combination of selected attributes according to appropriate similarity measures. It is also worth considering that POI locations from LBSN are often recorded by GPS positioning via smartphones and they can be inaccurate to tens of meters. Therefore, the more attribute will be involved in comparison, the more correct the matching will be.

Because of Foursquare dataset does not contain content information about POIs, that is required by our algorithm, we propose to conflate the POIs based on multiple attributes from the Foursquare dataset using the Yelp API [5]. It should be noted, that instead of the Yelp API it would be possible to use available Yelp dataset, but the problem is that both datasets cover different geographical region to varying degrees and do not contain a sufficient number of overlapping attributes.

[5]`https://www.yelp.com/fusion`

As a rule, the process of combining two data sources must be performed on overlapping attributes. The Foursquare dataset and Yelp API have several overlapping attributes that we used as input to match POI:

Geographic Location In POI matching, it may seem that the geographic distance between two locations is a strong indicator of the match accuracy. However, the location coordinates are subject to the same contribution errors that are present in any of the other attributes. Given the uncertainty of mobile positioning systems and the systematic errors inherent to GPS positioning, it is not uncommon to detect significant discrepancies in the geographic coordinates of the same location derived from two applications [41]. For example, the average distance between two POIs in our matched set is 72.9 meters with a maximum difference of 689.1 meters. To improve the accuracy of coordinates, we use the Foursquare API[6] to correct it. Fortunately, the Foursquare dataset contains venue ids that correspond to venues IDs in Foursquare API. After that, the average distance between two POIs in our matched set became 39.9 meters with a maximum difference of 307.5 meters.

Venue Name To compare two names between Foursquare and Yelp venue we use the Levenshtein distance as similarity measure. The Levenshtein distance between two strings is the minimum number of edit operations (additions, deletions or substitutions) required to transform one word into the other. The fewer changes required, the smaller the editing distance and the more similar the names of the two POIs. This similarity measure between two names of venues will be denoted as $Lev_{name}(a, b)$. Originally, Foursquare dataset did not contain the name of venue, so we use the Foursquare API to extend the existing dataset (merging was performed by venue ID).

Venue Category To improve the matching quality, the categories similarity between venues should be also taken into account. Each venue in the Foursquare contain short category description, for example ”Restaurant” or ”Food & Drink Shop”. The categories in Yelp similar, but still are different. For example, in Yelp it will be ”Coffeeshops” instead of ”Coffee Shop” in Foursquare. To compare categories, we split all categories names onto words (e.g., ”coffee” and ”shop” or ”food”, ”drink” and ”shop”) and check if category in Yelp contains these subwords. For example, we check if ”coffee” and ”shop” are contained in ”Coffeeshops”. The word ”Coffee” is contained and this word is removed from string and remained ”shops” is compared with ”shop”. The categories similarity between venues a and b will be denoted as $Cat(a, b)$.

[6] `https://developer.foursquare.com/places-api`

The process of merging each venue in the Foursquare dataset with the Yelp API can be described as follows:

1. *Request to Yelp API*: The Yelp API provide the endpoint for searching venues, that returns up businesses based on the provided search criteria. For example for venue with GPS coordinates (lat, lan) we call the following endpoint:

   ```
   GET https://api.yelp.com/v3/businesses/search
   ?latitude=lat
   &longitude=lan
   &radius=300
   &limit=20
   ```

   It must return the 20 closest venues to the given coordinates in the suggested search radius (in meters).

2. *Calculation of similarity*: For all returned items we compute distance, similarity of name and categories using the following weighted model:

   $$Sim(a, b) = \lambda_1 \cdot Dist(a, b) + \lambda_2 \cdot Lev_{name}(a, b) + \lambda_3 \cdot Cat(a, b)$$

   where a is the target venue in the Foursquare dataset, b is the returned venue from Yelp API, $\lambda_1, \lambda_2, \lambda_3$ are regularization parameters to control the impact of each similarity measure.

3. *Finding the best match*: If venue with the highest similarity value satisfies a pre-specified matching threshold, a match is found. If no, the venue is marked as unmatched.

To evaluate the validity of the proposed model, we randomly select 200 POIs in the Foursquare dataset. Selected venues consisted of a name, geographic coordinates and at least one category tag. Using the Yelp API, we request for the same number of POI and save all returned responses. Then we manually compare returned responses from Yelp to each target POI. For 200 POI only 170 positive matches were found, which is understandable because not all venues from Foursquare are presented in Yelp and vice versa. Then we use our proposed model and match the same POIs. The proposed model correctly matched 96% of the 170 POIs, that produce great match accuracy and confirm the validity of the proposed model. The main reason for not finding a match was that the distance between matched POI often differed significantly across providers.

Chapter 4

Experiments and evaluation

In this section, we design and conduct several experiments on real datasets to compare the recommendation quality of the proposed content-aware matrix factorization method with some state-of-the-art recommendation techniques. Specifically, the design of the experiments aims to address the following questions:

1. How does the control parameters (β, k and so on) impact the quality of recommendation? The goal is to understand the roles/weights of the parameters in obtaining optimal recommendations. See Section 4.3 for details.
2. How is the effectiveness of the proposed algorithm compared with other state-of-the-art collaborative filtering techniques? In Section 4.4, we intend to explore the feasibility and necessity of integrating content information into matrix factorization approaches to further enhance the recommendation quality.
3. How well does our approach deal with cold start item, i.e., with POIs that don't have a lot of ratings or check-in records? In Section 4.5, we explore the impact of leveraging additional content information on the cold start problem.

In the following, we'll look at all of these questions, but first we'll discuss interesting implementation details and evaluation metrics, used in our experiments.

4.1 Implementation details

In our experiments, we use two real-world datasets described in Chapter 3. The first dataset is the Yelp data containing ratings for POIs from Las Vegas, Phoenix, Toronto, and Scottsdale cities. These datasets will be used by our first proposed content-aware SGD algorithm for explicit feedback. The other dataset is the extended Foursquare data containing check-in data made in New York City. This dataset will be used by the second content-aware ALS algorithm for implicit feedback.

For each user, we randomly select 30 % of her visiting locations as testing data (also referred to as ground truth) to evaluate the performance of different algorithms. The remaining portions from each user constitute a training set for learning the parameters of the proposed model. Based on the training sets, we construct a user-POI rating matrix R with 111,565 nonzero entries and check-in frequency matrix C with 20,804 nonzero entries for Yelp and Foursquare respectively, which will be used in the POI recommendation. The sparsity of R is 99.46 % and 99.315 % of matrix C.

Almost all of the code is written in pure Python, except content-aware weighted matrix factorization for implicit feedback, that requires a very space-consuming full-matrix inverse operation. We use Matlab[7] to implement as the inverse operation there is better optimized. To store sparse matrices R and C we use SciPy[8] 2-D sparse matrix package, that allows us to perform complex matrix computations. For different matrix manipulation, we use NumPy[9] library, that provides an abundance of useful features for operations on n-arrays and matrices in Python. Also, we use Matlab Parallel Computing Toolbox and Python Multiprocessing module[10] for parallelization of computations.

All our experiments are conducted using PC with an Intel Core i5 2GHz Processor, 16GB memory and macOS operating system.

4.2 Evaluation metrics

To measure the recommendation quality of the two proposed methods for explicit and implicit feedback, we choose two different approaches. For the first algorithm for explicit feedback, where a user specifies preference using ratings, we use two popular metrics: Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) to measure the prediction accuracy. These metrics are based on the difference between the rating that you predicted for an item and the rating that a user gave that item. Formally,

$$MAE = \frac{\sum_{i=1}^{T} |r_i - \hat{r}_i|}{T}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{T} |r_i - \hat{r}_i|^2}{T}}$$

[7] `https://www.mathworks.com/products/matlab.html?s_tid=hp_products_matlab`

[8] `https://docs.scipy.org`

[9] `https://www.numpy.org`

[10] `https://docs.python.org/3.7/library/multiprocessing.html`

where r_i and $\hat{r}_i$ are the real rating and the corresponding prediction, respectively, and T denotes the total number of predictions generated for all active users. The accuracy is evaluated over entries in the test dataset. For the above equations, the lower the MAE or RMSE value, the better the recommendation algorithm. The differences between these metrics are that the RMSE is in units of ratings, rather than in units of squared ratings like the MSE. Also in contrast with MSE, the RMSE tends to disproportionately penalize large errors because of the squared term within the summation [6].

Evaluation of implicit-feedback recommendation method, however, requires other appropriate measures. It is important to realize that we don't have any preference data and reliable feedback from the user regarding which locations are unloved, as not visiting a POI can stem from multiple different reasons. Instead, we only have an implicit metric (visiting frequency) that we use to obtain an estimate of that preference. We can still generate a set of predicted values, but those values are only meaningful to rank items to be recommended and the values themselves don't really matter. Thus, using the above-mentioned error measure metrics for explicit feedback is a bit pointless.

In implicit-feedback recommendation methods, the learned model is assessed by its capacity of finding the ground truth locations for each user among the top k ranked locations. So it makes sense to look into rank-based metrics. These include, for example, MAP@k, Precision@k, Recall@k, DCG@k, and the percentile-ranking metric. In our experiments, we will use three widely used rank-based metrics in the top k POI recommendation – Recall@k, Precision@k, and nDCG@k.

The metric Recall@k indicates what percentage of the user's visiting locations can emerge in the top k recommended POIs while the second metric Precision@k indicates what percentage of locations among the top k recommended POIs has been visited by the user. Formally,

$$Recall@k = \frac{1}{M} \sum_{u=1}^{M} \frac{|\mathbb{S}_u(k) \cap \mathbb{V}_u|}{\mathbb{V}_u}$$

$$Precision@k = \frac{1}{M} \sum_{u=1}^{M} \frac{|\mathbb{S}_u(k) \cap \mathbb{V}_u|}{k}.$$

where $\mathbb{S}_u(k)$ is the top k recommended POIs, $\mathbb{V}_u$ represents all visited locations of user and M is count of users.

The nDCG@k metric (normalized Discounted Cummulative Gain) is defined for each user as:

$$nDCG@k = \frac{DCG@k}{IDCG@k} = \frac{\sum_{i=1}^{k} \frac{2^{rel_i}-1}{log_2(i+1)}}{\sum_{i=1}^{r} \frac{2^{rel_i}-1}{log_2(i+1)}}$$

where rel_i refers to the graded relevance of the result ranked at the position i and r represents the count of relevant POIs (ordered by their relevance) in the test set up to position k. In other words, IDCG is the DCG value when the recommended POIs are ideally ranked [42]. We divide the raw DCG by this ideal DCG to get normalized NDCG value between 0 and 1. In our experiment, K is set to be 10.

For the above measures, the higher the value, the better the recommendation algorithm. All these metrics are computed individually for each user and then the average metric value of all users is reported.

4.3 Impact of control parameters to quality of recommendation

In the experiments, we set the learning rate α in SGD algorithm to a small value 0.005 to ensure the generalization accuracy. The parameter ε for checking the convergence condition we set to 0.0001 for all the datasets. Finally, we use the number of iteration $w_1 = 200$ for SGD algorithm and $w_2 = 10$ for second ALS algorithm to control the loop conditions of matrix factorization procedures. Figure 4.1a show the first algorithm, where after 200 rounds the value of the objective function J that we minimalize is still decreasing, but only by a small value (the curve is almost flattened) for each iteration afterward. So for the first algorithm, we will set the number of iteration w_1 to 200.

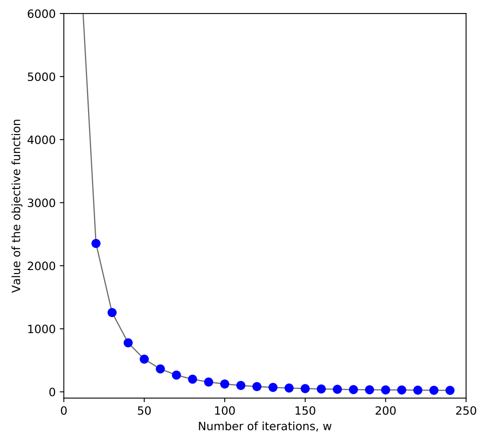
The second algorithm for implicit feedback, where we use ALS approach and trying to minimalize the objective function J^*, requires much less number of iteration to reach the convergence, but each iteration is very time-consuming (due to full-matrix inverse operation). You will see at Figure 4.1b, that after 10 iterations the algorithm reaches the convergence of the cost function $J*$ and each further value is no longer changing or the change is quite small. So for the second algorithm, we will set the number of iteration w_2 to 10.

In the following, we will explore the impact of control parameters of our algorithms to quality of recommendation and tune them based on the training set to find the optimal values. Subsequently, we will use the best values of parameters during comparison with other approaches in Section 4.4.

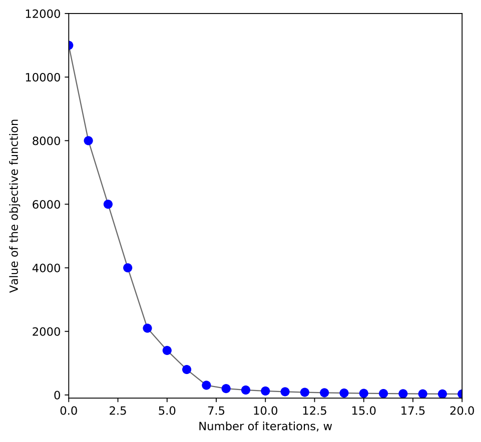
4.3.1 Impact of control parameter λ

In our proposed method, the parameter λ is used as a regularization term, $\frac{\lambda}{2}(||U||^2 + ||V||^2)$, that added to the objective function to avoid overfitting. To select the most appropriate regularization parameter λ we use the holdout method, where values of λ from 0.001 to 1 are tested over the training set and the value of λ that provides the highest accuracy is then used on the testing set. Another parameter K was set to 50 and the parameter β was set to 0.1.

Figure 4.2 reports the impacts of parameter λ on MAE and RMSE in the first SGD algorithm for explicit feedback. You can see, that the values of λ



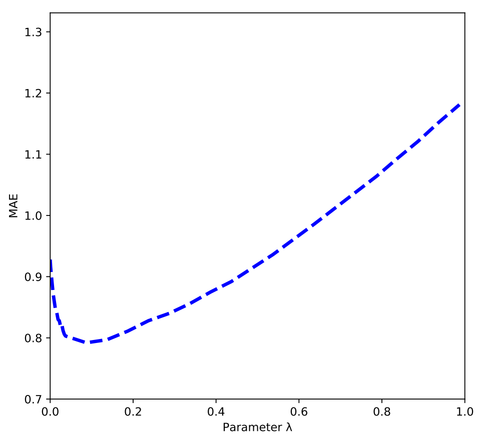
(a) SGD algorithm based on explicit rating



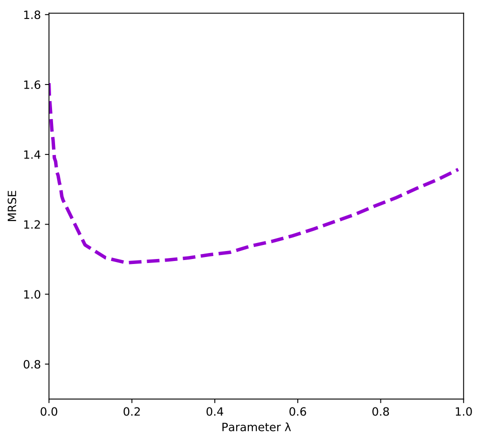
(b) ALS algorithm based on implicit feedback

Figure 4.1: Impact of the number of iterations on convergence of cost function J on the training dataset. Value of J is computed on each iteration.

have a significant impact on the recommendation quality. The curves of MAE and RSME show similar change trends: as the λ increases, the values of MAE and RMSE firstly drop down, but after the parameter λ reaches a certain threshold, the metric values begin to increase as the λ increases, which means that the performance degrades when λ is too large. Algorithm achieves the best performance MAE=0.797 and MRSE =1.104 when λ is around 0.135.



(a) Impact λ on MAE



(b) Impact λ on MRSE

Figure 4.2: Impact of λ on prediction accuracy of SGD algorithm based on explicit rating

Figure 4.3 reports the impacts of λ on Recall@k, Precision@k and nDCG@k metrics (with k set to 5 and 10) in the second ALS algorithm for implicit feedback. You can see, that unlike the first algorithm, the recommendation performance of the second algorithm is not very sensitive to the regularization coefficient λ and does not strongly influence recommendation quality. This can be explained by the fact that all negative examples (unvisited locations) are taken into account.

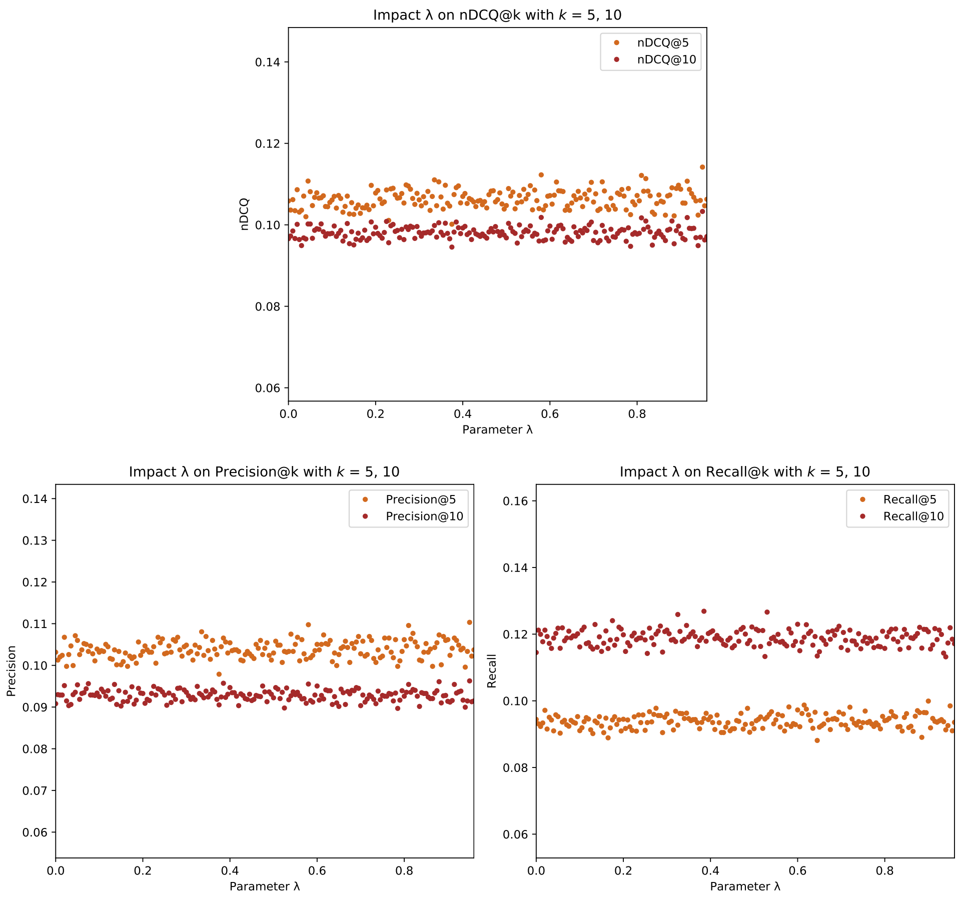


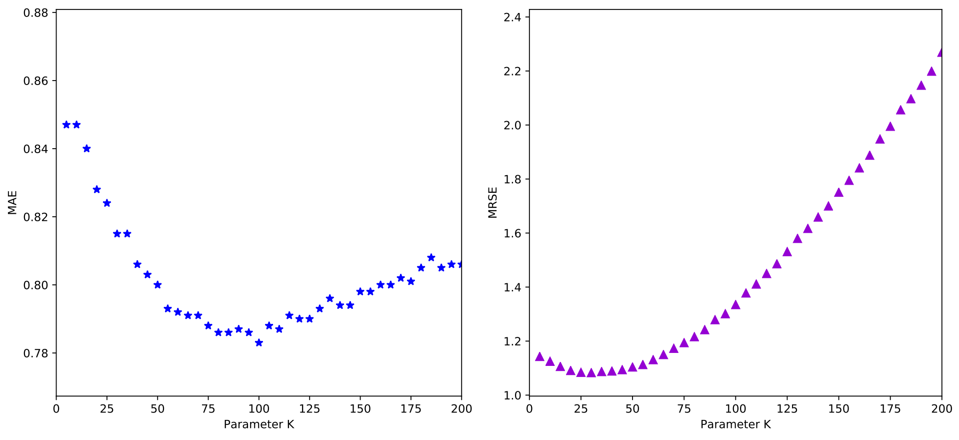
Figure 4.3: Impact of λ on prediction accuracy of ALS algorithm based on implicit rating

However, if highlight the best values, the second ALS algorithm based on implicit feedback achieves the best performance Recall@5=0.099, Precision@5=0.11 and nDCG@5=0.114 when λ is around 0.95 and Recall@10=0.126, Precision@10=0.096 and nDCG@10 when λ is around 0.103.

4.3.2 Impact of control parameter K

The parameter $K \ll min(m, n)$ denotes the dimension of latent feature vectors U and V, that is, the number of hidden features in the model. It is another important parameter in our proposed method. To assess the impact of parameter K on the recommendation quality we set K to values from 5 to 200 with a step of 5. Parameter λ was set to 0.135 and the parameter β was set to 0.1.

The experimental results for the first algorithm are plotted in Figure 4.4. You can see that as K increases, the values of MAE decrease to reach K a certain threshold and then begin to increase. The values of MRSE behave a little differently – at first, it decreases a little to reach K the value 35 and then starts to increase quite quickly. To sum up, the first algorithm achieves the best performance MAE=0.783 when K is around 100 and MRSE=1.083 when K is around 35.



(a) Impact of K on MAE (b) Impact of K on MRSE

Figure 4.4: Impact of K on prediction accuracy of SGD algorithm based on explicit rating

The experimental results for the second algorithm are plotted in Figure 4.5. You can see that as K increases, the values of Recall, Precision and nDCG metrics increase to reach K a certain threshold and then begin to decrease. So, increasing K does not improve the performance of the second algorithm after K exceeds a certain threshold. To sum up, the second algorithm achieves the best performance Precision@5=0.106, Recal@5=0.096 and nDCG@5=0.108 when K is around 30 and Precision@10=0.093, Recal@10=0.121 and nDCG@10=0.099 when K is around 70.

Based on the intuition that larger values of K may represent more preferences by latent feature vectors, the quality of the recommendation should be increase as K increase. However, Figures 4.4 and 4.5 show that continually

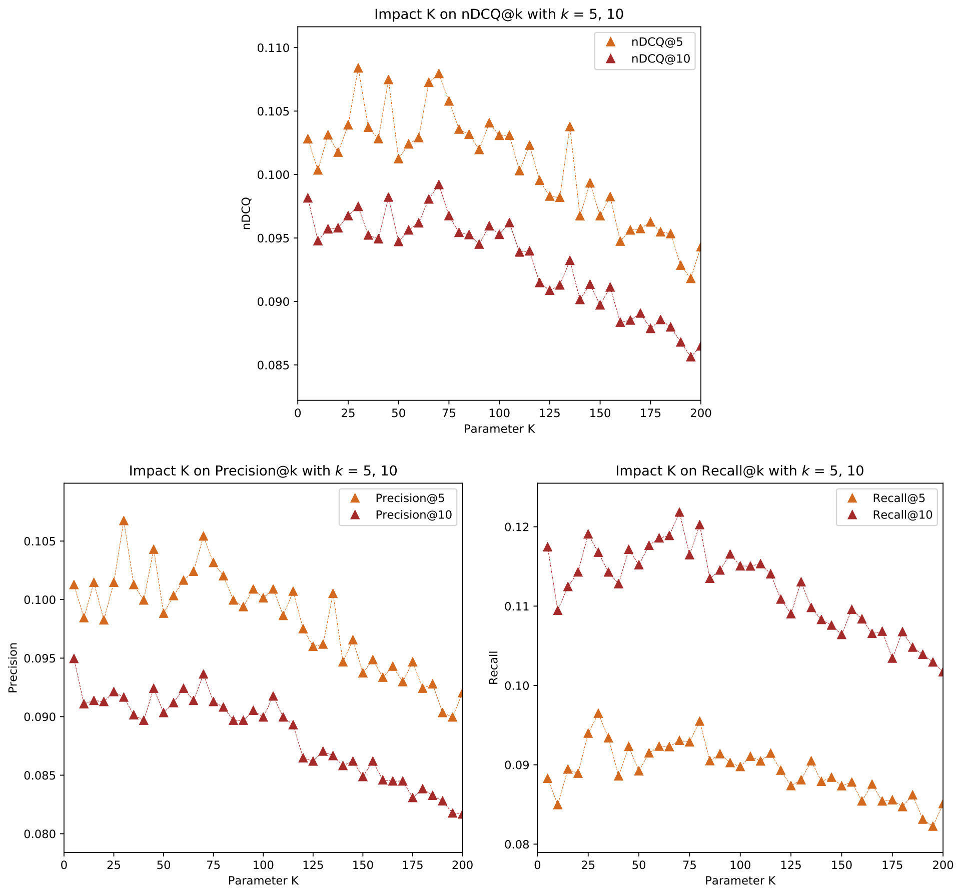


Figure 4.5: Impact of K on prediction accuracy of ALS algorithm based on implicit rating

increasing the dimension of the latent feature vector does not improve performance after K exceeds a certain threshold. A possible reason is that the latent feature vectors with certain dimension K are enough to characterize the preferences of the users and items and increasing the K will introduce a lot of noise into the objective function, which will lead to a deterioration in the quality of the recommendations. This may explain that the effect of noise on MRSE values (see Figure 4.4b) is quadratic because it uses a square term in the summation.

4.3.3 Impact of function μ in weighting matrix

Recall, that in the second algorithm for implicit feedback we are using the weighted matrix factorization approach, where the weighted matrix W is adding to the objective function, that defined as follows:

$$w_{ui} = \begin{cases} \mu(c_{ui}) + 1, & \text{if } c_{ui} > 0 \\ 1, & \text{otherwise} \end{cases}$$

where $\mu(c_{ui}) > 0$ is a monotonically increasing function.

Using this weighted matrix, all non-visited locations are considered as negative examples and the weights to all negative examples are assigned the same value, i.e., 1. As for positive examples, to all visited POIs a greater weight (with respect to c_{ui}) is assigned and the rate of weight increase is controlled by the function $\mu(c_{ui})$.

So, the function $\mu(c_{ui})$ plays an important role in the weighted matrix factorization approach. To select the appropriate function and evaluate its impact on the recommendation quality, we conduct the experiments by considering the following variations of the increasing function μ:

$$\mu_1(c_{ui}) = z * c_{uj}$$

$$\mu_2(c_{ui}) = log(1 + c_{ui} * (10^{\varepsilon}))$$

In this experiment, the parameter λ was set to 0.1, the parameter K was set to 30 and the parameter β was set to 0.1. The experimental results for the first function μ_1 are plotted in Figure 4.6. In this function, the rate of increase is controlled by the constant z. As you can see, setting $z = 10$ was found to produce good results, namely Precision@5=0.09, Recall@5=0.079 and Precision@10=0.076, Recall@10=0.101.

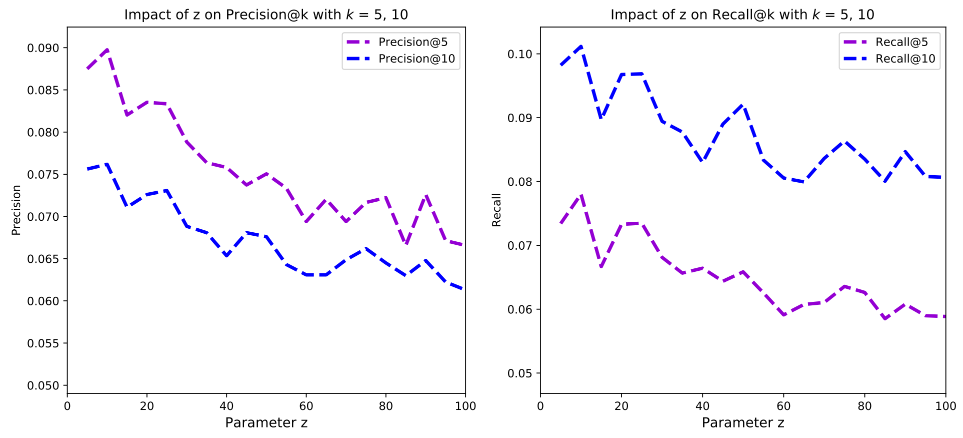


Figure 4.6: Impact of function $\mu_1(c_{ui}) = z * c_{uj}$ on prediction accuracy of ALS algorithm based on implicit rating

The experimental results for the second function μ_2 are plotted in Figure 4.7. In this function, the rate of increase is controlled by the constant 10^{ε} and logarithmic function. As you can see, setting $\varepsilon = 2$ produce the best performance, namely Precision@5=0.093, Recall@5=0.079 and Precision@10=0.077, Recall@10=0.105.

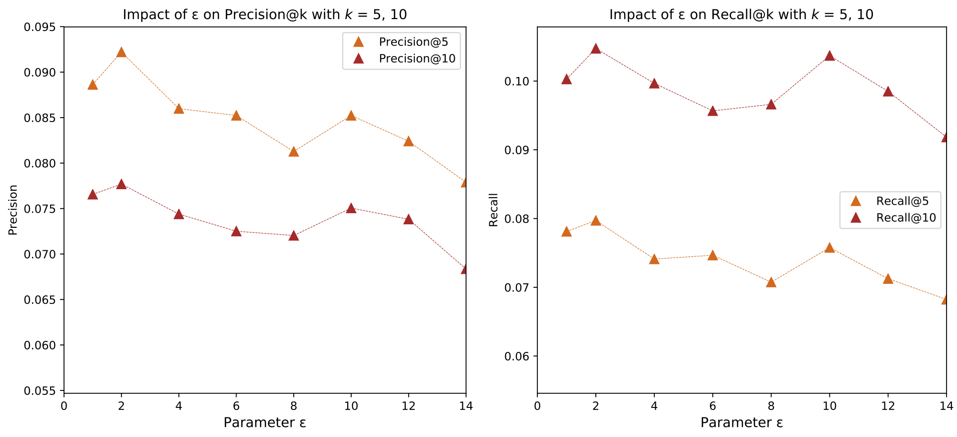


Figure 4.7: Impact of function $\mu_2(c_{ui}) = log(1 + c_{ui} * (10^{\varepsilon}))$ on prediction accuracy of ALS algorithm based on implicit rating

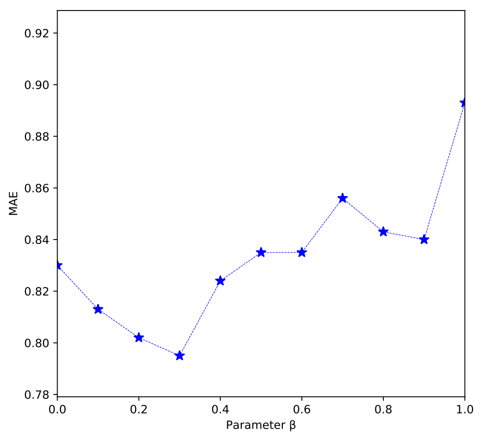
The best performance results of both functions μ_1 and μ_2 are similar, but the function μ_2 produces higher values, so in the following we will use the function $\mu_2(c_{ui}) = log(1 + c_{ui} * (10^{\varepsilon}))$ with setting $\varepsilon = 2$.

4.3.4 Impact of control parameter β

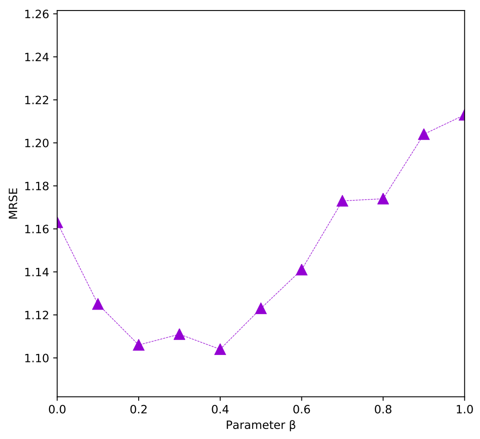
In the proposed method, the parameter β plays an important role and controls the influence of POI attribute information in the factorization process – a larger value of β indicates that we put more weights into the attribute information. To assess the impact of parameter β on the recommendation quality we conduct the experiments by changing the values of β to values from 0 to 1 with a step of 0.05. Parameter λ was set to 0.135 and the parameter K was set to 35 and 30 for the first and second algorithm respectively.

Note, that when parameter β is set to 0, our method degrades to the baseline regularized matrix factorization approach, and setting the parameter β to 1 makes the latent feature vectors close to the direct neighbors.

The experimental results for the first algorithm for explicit feedback are plotted in Figure 4.8, that reports the impacts on MAE and RMSE. You can see, that the curves of MAE and RSME show similar change trends – as β increases, the values of MAE decrease (the recommendation quality improves) to reach β a certain threshold and then begin to increase. So we can conclude, that the values of β have a noticeable impact on the recommendation quality of the first algorithm, but the performance degrades when β is too large. To sum up, the first algorithm achieves the best performance MAE=0.795 and MRSE=1.11 when β is around 0.25.



(a) Impact of β on MAE



(b) Impact of β on MRSE

Figure 4.8: Impact of β on prediction accuracy of SGD algorithm based on explicit rating

The experimental results for the second algorithm for implicit feedback are not shown because all curves show similar change trends as for the first algorithm - as β increases, the metric values decrease (the recommendation quality improves) to reach β a certain threshold and then begin to increase. We evaluated impacts on Precision, Recall, and nDCG. The second algorithm achieves the best performance Precision@5=0.097, Recal@5=0.091 and nDCG@5=0.101 when β is around 0.15 and Precision@10=0.091, Recal@10=0.11 and nDCG@10=0.099 when K is around 0.1.

Note, that the first algorithm shows the best performance when $\beta = 0.25$ and the second algorithm when β is around 0.1. It can be explained by the fact, that the second algorithm uses the extended Foursquare data, that contain less attribute and categories information because it was combined with Yelp dataset and not always all content information was found during matching.

From the experimental results, it can be concluded that very large values or very small values of β hurt the recommendation quality. When the parameter β is too large, POI attribute information will dominate the learning process and make the latent feature vectors close to the direct neighbors [4]. A small value of β causes our method to degrade to the baseline regularized matrix factorization approach. In the following, β will be set to 0.25 and 0.1 for the first and second algorithms respectively.

4.4 Comparison with other approaches

To evaluate the performance of the proposed method, we choose the following state-of-the-art collaborative filtering approaches for comparison:

RSVD Regularized SVD approach denoted as RSVD is the matrix factorization approach which is widely used because of its good scalability and high recommendation quality. This method only utilizes a user-item rating matrix R to generate a prediction. The objective function of this approach is defined in Equation 2.1 and this method is used as the baseline approach in our first algorithm for explicit feedback (by setting the parameter β to 0, we obtain the RSVD algorithm).

WMF Weighted matrix factorization approach denoted as WMF is proposed by Yifan Hu et al. [40]. In this algorithm, users and POIs are mapped into a joint latent space by approximating a user-POI 0/1 rating matrix (each 0/1 entry indicates whether a user has visited the POI) in a weighted manner [37]. This method is suitable for processing implicit feedback and our second algorithm based on check-in data exploit this approach as a basis (by setting the parameter β to 0 in Equation 2.5, we obtain the WMF algorithm).

SVD++ The SVD++ algorithm is an extension of SVD and was proposed by Yehuda Koren [43]. This approach is an integrated model that combines the neighborhood and the latent factor models. It uses an implicit feedback matrix, that can be derived from the rating matrix or other forms of implicit feedback.

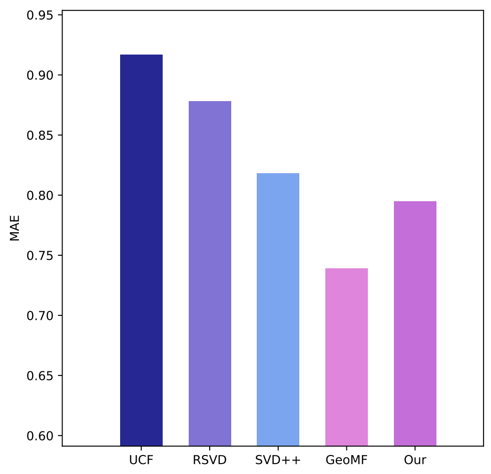
UCF k-NN User-based Nearest Neighbor algorithm denoted as UCF k-NN is a basic collaborative filtering approach. This method utilizes only a user-item rating (or check-in matrix) for computing user-user similarity to generate recommendations subsequently. This method is described in detail in Section 1.2.1.

GeoMF GeoMF [37] is the state-of-the-art method for POI recommendation. This method is described in detail in Section 1.5.1.

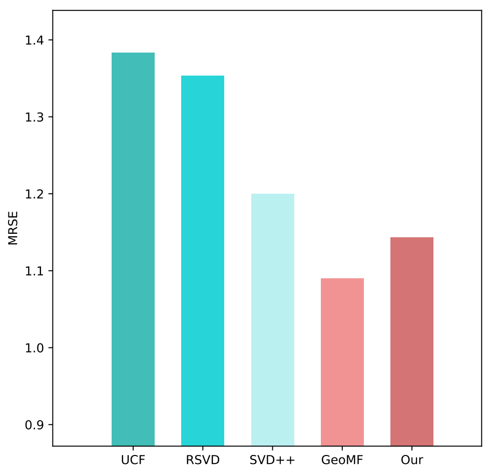
The SVD++ and k-NN algorithms were implemented using Surprise [44] library, that has a set of built-in algorithms, the GeoMF approach was implemented using source code available at [45].

To make a fair comparison, we set the common parameters to be identical parameter values in the all methods. For all involved recommendation algorithms, we set $\lambda = 0.1$ and the learning rate α is set to 0.005. The control parameter β is set to 0.1. Finally, we use $\varepsilon = 0.0001$ and the number of iteration $w_1 = 200$ and $w_1 = 10$ to control the loop conditions of matrix factorization procedures.

The results of comparison the above selected recommendation algorithms with our first algorithm for explicit feedback are plotted in Figure 4.9. We don't evaluate the WMF algorithm, because it assumes using the implicit feedback instead of explicit, so the comparison with our algorithm would not provide any useful information for us (the first algorithm uses the dataset with explicit feedback, i.e., ratings).



(a) Comparison of MAE



(b) Comparison of MRSE

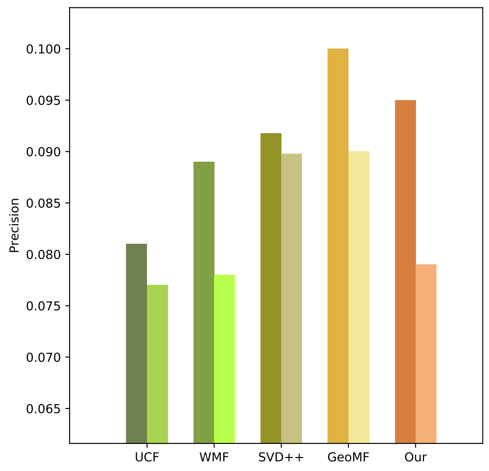
Figure 4.9: Comparison the selected recommendation algorithms with our first algorithm for explicit feedback

In Figure 4.9a you can see, that in terms of MAE, the first algorithm outperforms UCF, RSVD, SVD++, but GeoMF gives the better MAE values on 7.5% than our algorithm. In terms of MRSE, Figure 4.9b shows the similar results. GeoMF algorithm is one of the state-of-the-art methods for POI recommendation, that integrates geographical influence by modeling users' activity regions and the influence propagation on geographical space [42].

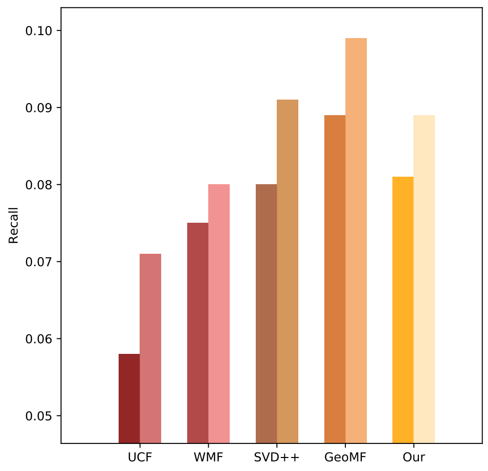
The results of comparison the above selected recommendation algorithms with our second algorithm for implicit feedback are plotted in Figure 4.10.

We don't evaluate the RSVD algorithm, because it assumes using the explicit feedback instead of explicit. In addition, in Section 2.2.1 we explain, why the using this approach doesn't make sense in terms if implicit feedback. So the comparison RSVD with our second algorithm would not provide any useful information for us.

In Figure 4.10a you can see, that in terms of Precision@5, our second algorithm outperforms UCF, WMF, SVD++, but GeoMF outperforms the second algorithm of implicit feedback by 5%. As for Precision@10, the second algorithm outperforms UCF, WMF but GeoMF and SVD++ are superior to our second algorithm of implicit feedback by about 12%. In terms of Recall, Figure 4.10b shows similar results. In terms of Recall@5, our second algorithm



(a) Comparison of Precision@k, k=5 (left bar) and 10 (right bar)



(b) Comparison of Recall@k, k=5 (left bar) and 10 (right bar)

Figure 4.10: Comparison the selected recommendation algorithms with our second algorithm for implicit feedback

outperforms UCF, WMF, SVD++, but GeoMF outperforms the second algorithm by 9.8% . As for Recall@10, the second algorithm outperforms UCF, WMF but GeoMF and SVD++ are superior to the second algorithm of implicit feedback by 11% and 2.2% respectively.

Figure 4.11 shows the comparison results of nDCG. You can see, that in term of nDCG@5, only GeoMF outperforms the second algorithm of implicit feedback by around 7%. As for nDCG@10, WMF and our second algorithm are superior to the others.

This observation confirms the assumption that the use of POI content information can improve the quality of recommendations. However, almost always GeoMF algorithm is superior to our proposed methods. The GeoMF algorithm integrates geographical influence by modeling users' activity regions and the influence propagation on geographical space [42]. Therefore, it makes sense to consider using the geographical influence in our approach to improve the recommendation quality.

4.5 Performance on Cold Start Items

One advantage of the collaborative filtering approach is that it does not require additional information about users or items. Thus, it is able to recommend an item without understanding the element itself [46]. However, this benefit is also the root cause of the so-called cold start problem, which refers to the general difficulty in performing collaborative filtering for users and items

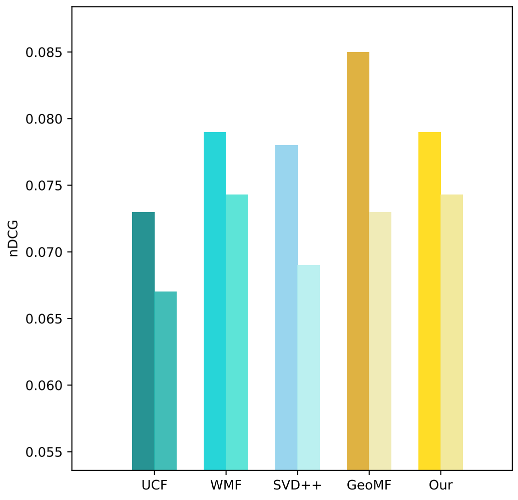


Figure 4.11: Comparison of nDCG@k (k=5 left bar and k=10 right bar) with the selected recommendation algorithms and our second algorithm for implicit feedback

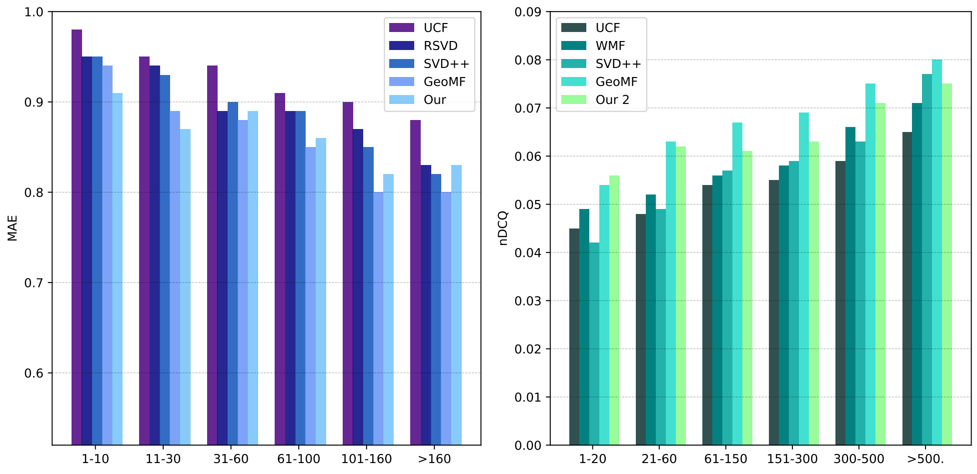
that are relatively new. New users who haven't rated many items have no information about them to be compared with other users. Like for new users, no user ratings are available for the new item. If we consider POIs which are rated by users less than 20 times or visited less than 30 times as cold start items, 38.97%, and 40.2% are cold start items in first Yelp dataset and extended Foursquare dataset, respectively. It is quite a large percentage.

Because we use additional content information for construct similarity matrix between POIs, it can help to deal with the cold start item issue in recommender systems. So, in this section, we will evaluate the effectiveness of our approach to coping with the cold start item problem.

We group POIs according to the number of observed ratings and check-ins on POIs in the training set, and then compare the values of metrics of different POIs groups with other selected recommendation algorithms. We consider the following groups categories for ratings: 1-10, 11-30, 31-60, 61-100, 101-160, >160 and the following categories for check-ins: 1-20, 21-60, 61-150, 151-300, 300-500 and >500.

The results of the comparison of selected recommendation approaches with our first algorithm for explicit feedback are plotted in Figure 4.12. You can see, that the first algorithm is able to generate better recommendations than other algorithms when the POIs have few observed ratings (11-30). As more observed ratings are given (>30), the improvement of our proposed approach gradually reduces.

The results of the comparison of selected algorithms with our second algorithm for implicit feedback are plotted in Figure 4.12b. The figure shows similar change trends as for the first algorithm - the algorithm is able to generate better recommendations than other algorithms when the POIs have few check-ins count(1-20). As more check-ins are given (>21), the improvement of our proposed approach gradually reduces.



(a) Comparison of MAE with the first algorithm

(b) Comparison of nDCG@5 with the second algorithm

Figure 4.12: Comparison the selected recommendation algorithms with our algorithms

These observations indicate that our proposed algorithms can cope with cold start item problem and the improvement is directly related to the use of additional content information, such as POI attribute and categories information.

Conclusion and future works

With the availability of the vast amount of users and Location-based social networks, the problem of POI recommendations has been widely studied and received significant research attention in the last seven years, and many approaches have been suggested. While previous works of POI recommendation mostly focused on investigating the spatial, temporal, and social influence, the use of additional content information has not been directionally studied.

In this paper, we proposed the content-aware matrix factorization method based on incorporating POI attribute and categories information to overcome the cold start item problem, and consequently improve the quality of recommendation. We proposed two variants of the algorithm that can work with explicit and implicit feedback. The attribute and categories information of a POI was collected from existing datasets and will be used to measure the similarity between two POIs. These similarity values were used to regularize the matrix factorization by adding item relationship regularization term to the objective function of matrix factorization [4].

Experimental results show that the proposed method improves the quality of recommendation and can effectively cope with the so-called problem. The proposed method outperforms most state-of-the-art collaborative filtering algorithms, such as RSVD, WMF, SVD++, and UCF k-NN. Of all the algorithms tested, only GeoMF algorithm has surpassed our approach. Therefore, it makes sense to consider using the geographical influence on users' check-in or rating behaviors based on the assumption that users tend to visit nearby locations within a radius of activity regions. Furthermore, it would be interesting to investigate the recommendation effect of content information compared to other information, such as temporal or social information (e.g., take into account the user social networking relations).

Also, in our work, we use the simple similarity measure between attributes. Because some of the attributes are in categorical structure, it would be fine to consider some similarity measure, that reflects the relationship between categorical data. For example, in the work [47], the Coupled Object Similarity measure is used to measure the similarity between items based on the item-attribute information. Besides, this measure was successfully implemented in [4] to measure the similarity between movie attributes. And finally, it would be interesting to investigate other types of content information available in LBSNs, for example, sentiment indications of textual reviews.

Bibliography

[1] Yu, Y.; Chen, X. A Survey of Point-of-Interest Recommendation in Location-Based Social Networks. *Association for the Advancement of Artificial Intelligence*, 2015, [cit. 2019-02-23].

[2] Weber, H.; Novet, J. Foursquare by the numbers: 60M registered users, 50M MAUs, and 75M tips to date. [online], 2018, [cit. 2019-02-23]. Available from: `https://venturebeat.com/2015/08/18/foursquare-by-the-numbers-60m-registered-users-50m-maus-and-75m-tips-to-date/`

[3] 10 Things You Should Know About Yelp. [online], 2018, [cit. 2019-02-23]. Available from: `https://www.yelp.com/about`

[4] Yu, Y.; Wang, C.; et al. Attributes Coupling based Item Enhanced Matrix Factorization Technique for Recommender Systems. *IEEE Transactions on knowledge and data engineering*, 2014.

[5] Adomavicius, G.; Tuzhilin, A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on knowledge and data engineering*, 2005, [cit. 2019-02-24].

[6] Aggarwal, C. *Recommender Systems: The Textbook*. Springer International Publishing, IBM T.J. Watson Research Center Yorktown Heights, NY, USA, 2016, ISBN 9783319296579.

[7] Sarwar, B. M. Memory-based Collaborative Filtering Algorithms. *The Tenth International World Wide Web Conference*, 2001, [cit. 2019-02-24]. Available from: `http://wwwconference.org/proceedings/www10/papers/519/node7.html`

[8] Resnick, P.; Iacovou, N.; et al. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, 1994, pp. 175–186.

[9] Breese, J.; Heckerman, D.; et al. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 1998, pp. 43–52.

[10] Sarwar, B.; Karypis, G.; et al. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, 2001, pp. 285–295.

[11] Linden, G.; Smith, B.; et al. Amazon. com recommendations: Item-to-item collaborative filtering. In *Internet Computing, IEEE*, 2003, pp. 76–80.

[12] Gershman, A.; Meisels, A. A Decision Tree Based Recommender System. *Department of Computer Science, Ben-Gurion University of the Negev*, 2010.

[13] Sarwar, B.; Karypis, G.; et al. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce*, 2000, pp. 158–167.

[14] Lin, W.; Alvarez, S.; et al. Efficient adaptive-support association rule mining for recommender systems. In *Data mining and knowledge discovery*, 2002, pp. 83–105.

[15] Hofmann, T. Latent semantic models for collaborative filtering. In *ACM Transactions on Information Systems (TOIS)*, 2004, pp. 89–115.

[16] Xue, G.; Lin, C.; et al. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, 2005, pp. 114–121.

[17] Pinela, C. Content-Based Recommender Systems [online]. November 2017, [cit. 2019-02-27]. Available from: `https://medium.com/@cfpinela/content-based-recommender-systems-a68c2aee2235`

[18] Pazzani, M. A Framework for Collaborative, Content-Based and Demographic Filtering. *Department of Information and Computer Science*, 1999, [cit. 2019-02-28]. Available from: `http://www.nargund.com/gsu/mgs8040/resource/dss/aireview.pdf`

[19] Claypool, M.; Gokhale, A.; et al. Combining content-based and collaborative filters in an online newspaper. *Proc. ACM SIGIR '99 Workshop Recommender Systems: Algorithms and Evaluation*, 1999.

[20] Billsus, D.; Pazzani, M. User modeling for adaptive news access. In *User Modeling and User-Adapted Interaction*, 2000, pp. 147–180.

[21] Nandi, M. Recommender Systems through Collaborative Filtering. *Domino Data Science Blog*, July 2017, [cit. 2019-03-09]. Available from: `https://blog.dominodatalab.com/recommender-systems-collaborative-filtering/`

[22] Agarwal, A.; Chauhan, M. Similarity Measures used in Recommender Systems: A Study. *International Journal of Engineering Technology Science and Research*, June 2017, [cit. 2019-03-09]. Available from: `https://pdfs.semanticscholar.org/943a/e455fafc3d36ae4ce68f1a60ae4f85623e2a.pdf`

[23] Bin, W. Comparison of User-Based and Item-Based Collaborative Filtering. May 2018, [cit. 2019-03-22]. Available from: `https://medium.com/@wwwbbb8510/comparison-of-user-based-and-item-based-collaborative-filtering-f58a1c8a3f1d`

[24] Gemulla, R.; Nijkamp, E.; et al. Large-scale matrix factorization with distributed stochastic gradient descent. In *ACM KDD Conference*, 2011, pp. 69–77.

[25] Jain, P.; Netrapalli, P.; et al. Low-rank matrix completion using alternating minimization. In *ACM Symposium on Theory of Computing*, 2013, pp. 665–674.

[26] Takacs, G.; Pilaszy, I.; et al. Matrix factorization and neighbor based algorithms for the Netflix prize problem. *ACM Conference on Recommender Systems*, 2008: pp. 267–274.

[27] Jaschke, R.; Marinho, L.; et al. Tag recommendations in folksonomies. *Knowledge Discovery in Databases (PKDD)*, 2007: pp. 506–514.

[28] R.Belland; Y.Koren. Lessons from the Netflix prize challenge. In *ACM SIGKDD Explorations Newsletter*, 2007, pp. 75–79.

[29] Ma, H.; Yang, H.; et al. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, 2008, pp. 931–940.

[30] Jamali, M.; Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the fourth ACM conference on Recommender systems*, 2010, pp. 135–142.

[31] Ye, M.; Yin, P.; et al. Location recommendation for location-based social networks. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2010, pp. 458–461.

[32] Ye, M.; Yin, P.; et al. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 2011, pp. 325–334.

[33] Yuan, Q.; Cong, G.; et al. Time-aware point-of-interest recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 2013, pp. 363–372.

[34] Liu, B.; Fu, Y.; et al. Learning geographical preferences for point-of-interest recommendation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 1043–1051.

[35] Gao, H.; Tang, J.; et al. Exploring temporal effects for location recommendation on location-based social networks. In *Proceedings of the 7th ACM conference on Recommender systems*, ACM, 2013, pp. 93–100.

[36] Cheng, C.; Yang, H.; et al. Fused matrix factorization with geographical and social influence in location-based social networks. In *AAAI Conference on Artificial Intelligence*, volume 12, 2012, pp. 17–23.

[37] Lian, D.; Zhao, C.; et al. Geomf: Joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 831–840.

[38] Li, X.; Cong, G.; et al. Rank-geofm: A ranking based geographical factorization method for point of interest recommendation. In *SIGIR*, 2015, pp. 433–442.

[39] Wanga, H.; Terrovitis, M.; et al. Location recommendation in location-based social networks using user check-in data. In *SIGSPATIAL*, 2013, pp. 374–383.

[40] Hu, Y.; Koren, Y.; et al. Collaborative Filtering for Implicit Feedback Datasets. In *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 263–272.

[41] McKenzie, G.; Janowicz, K.; et al. Weighted Multi-Attribute Matching of User-Generated Points of Interest. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013, pp. 440–443.

[42] Liu, Y.; ang Gao Cong, T.-A. N. P.; et al. An Experimental Evaluation of Point-of-interest Recommendation in Location-based Social Networkss. In *Proceedings of the VLDB Endowment*, volume 10, 2017, pp. 1010–1021, [cit. 2019-03-01].

[43] Koren, Y. Factorization Meets the Neighborhood: a Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 426–434.

[44] Hug, N. Surprise, a Python library for recommender systems. `http://surpriselib.com`, 2017.

[45] Liu, Y.; Pham, T.-A. N.; et al. An Experimental Evaluation of Point-of-interest Recommendation in Location-based Social Networks (full version). `http://spatialkeyword.sce.ntu.edu.sg/eval-vldb17/`, 2017.

[46] Su, X.; Khoshgoftaar, T. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[47] Wang, C.; Cao, L.; et al. Coupled nominal similarityy in unsupervised learning. In *CIKM*, 2011, pp. 973–978.

Appendix A

Acronyms

POI	Point-of-Interest
LBSN	Location-based social networks
GPS	Global position system
SVD	Singular value decomposition
PCA	Principal component analysis
MLE	Maximum likelihood estimation
SGD	Stochastic gradient descent
MAE	Mean Absolute Error
RMSE	Root Mean Squared Error
ALS	Alternative Least Squares

Appendix **B**

Contents of enclosed CD

```
readme.txt ...................... the file with CD contents description
src ....................................the directory of source codes
  wbdcm ..................................... implementation sources
  thesis .............the directory of LaTeX source codes of the thesis
text .........................................the thesis text directory
  thesis.pdf..........................the thesis text in PDF format
  thesis.ps............................the thesis text in PS format
```