Master of Science in Informatics at Grenoble
Master Mathématiques Informatique - spécialité Informatique
Artificial Intelligence and Web

Using Explicit and Implicit Feedback to Model Context for Proactive Recommendations

Sanjay-Kamath Ramachandra-Rao

22 June 2016

Supervised by M.-C. Fauvet & L. Goeuriot (LIG/MRIM - UGA)

Jury:
Prof. Eric Gaussier (external reviewer)
Prof. Nadia Brauner
Prof. Massih-Reza Amini
Prof. Catherine Berrut
Prof. Jérôme Euzenat

June 2016

Abstract

Data on the web are numerous and their number is increasing everyday, novel techniques are discovered everyday to use this data in productive manner. In spite of privacy concerns in using personal data for suspicious surveillance, some data are generated by people for public use. Such public generated data can be used for providing public news, accident reports, recommendations, seasonal sales, crowd behavior analysis, traffic updates, weather conditions, etc.

Availability of such data provides lot of scopes for various domains. Our work focuses on using these data for recommendations in travel domain. Traditionally, recommendations are delivered individually based on the ratings, likes (data explicitly given by users) or crowd presence, based on nearby locations, based on past visits (implicit data deduced by users' behaviors).

Context-aware recommendations use certain factors to deliver right recommendations at right situations. A context often refers to a situation which is described using some factors. The term *context* has a broad definition and is defined for different works in different ways. Context-awareness in travel recommendations can be seen as time-aware, location-aware, social-aware recommendations provided to the user. When a situation matches the predefined set of attributes (defined as context), the corresponding recommendations can be delivered to the user. This is useful for providing proactive recommendations - a new family of push mode recommendations. Our idea behind this work is to use different types of feedback such as crowd presence, ratings/likes about the Point-Of-Interests (POIs) for users during travel and provide time aware and location aware recommendations.

We use the open data available on the web, discuss problems involved in gathering right data and some existing techniques to generate such data in different ways. We propose a tensor factorization model with unified feedback of both explicit (ratings/likes etc.) and implicit (crowd presence, visit history) data from users about the venues they have visited in the past. Recommendations are evaluated separately, and together based on the feedback considered for the input.

Acknowledgements

I would like to thank my master thesis advisors Prof. Marie-Christine Fauvet and Dr. Lorraine Goeuriot for all the efforts they had to take to help me finish my master thesis. The door to their offices were always open whenever I ran into a trouble spot or had a question about my research or writing. They consistently helped me out in taking the right path for research, and steered me in the right the direction whenever I needed it.

I have put them into various troublesome situations such as writing overnight mails and expecting response, weekend short deadlines, finishing work late on friday nights, etc. but they helped me with their best efforts in all such situations and made me work more focussed and organised.

I would also like to thank my team members from MRIM team of LIG for providing valuable guidelines, comments and helping out with various technical difficulties I ran into, during my master 1 and master 2 thesis.

Finally, I must express my very profound gratitude to my parents and friends for providing me with unfailing support and continuous encouragement throughout my years of study in France and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Thank you.

Sanjay Kamath Ramachandra Rao

Contents

List of Figures

— 1 —
Introduction

1.1 Recommender Systems

Recommender Systems (RSs) are software tools and techniques which provide suggestions for useful items to users [RV97, Bur07, MR09, RRS11]. The recommendations (or suggestions) help a user to make decisions in buying items, listening to popular songs, visiting places, eating different cuisines, buying different clothes etc. Recommender systems can be applied to any domain which provides a list of choices for the user to choose from.

In general, recommender systems are supplementary software of a bigger system which help users make decisions by providing choices. To learn about the user's choice, these systems use the feedback given by her and her profile information to recommend the items which she might like. Recommendation of places differ from the recommendations for items such as movies, shopping sites, music etc. Because shopping websites, new songs, new books are less frequently used compared to places visited such as parks, restaurants, bars, or shopping malls when a user is traveling.

1.2 Recommendations in Travel

The recommendations of good places in a city for a newly visiting user, cannot be inferred from the past activities in his hometown. Especially when a user travels to a different country, the cultural places, specialties, local places will not be identified if the user's past data are the only input for recommending new places. Hence these data about the new city have to be collected from the users residing in that city or travelers who visited the city already, which can be a better source of data for new users to the system. With this assumption, there is a vast amount of data available and generated from the users in regard to places.

A simple recommendation example is shown in Figure 1.1 where a list of unranked items and the user data are the input to the recommender system. The output of a recommender system is a list of items, which is ranked and called recommendations.

Context-awareness is gaining attention in recommender systems since few years. Some works also state the necessity of contextual variables or factors in the traditional recommenders [AM06, Dey01]. Context-awareness tailors the recommendations to the right set of users at the right situations, rather than a generic set of recommendations at all temporal intervals. For

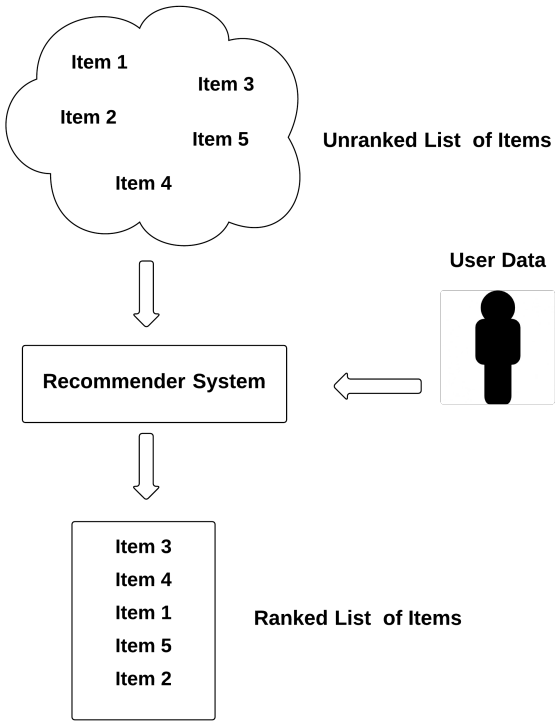

Figure 1.1 – Simple recommendation example

instance, a traveler looking for POI recommendations at nigh time will get a list of recommendations which might contain places closed at that time. Though the places recommended are popular in that vicinity, ignoring the opening hours of the place will not result in a suitable recommendation at that instance. This is an example for non context-aware recommendation. Hence considering *context* such as time, in a recommender system might leverage its accuracy.

Context could be seen as a multifaceted concept that has been studied across different research disciplines, including computer science (primarily in artificial intelligence and ubiquitous computing), cognitive science, linguistics, philosophy, psychology, and organizational sciences [AT11]. Since context has been studied in multiple disciplines, each discipline tends to take its own idiosyncratic view that is somewhat different from other disciplines and is more specific than the standard generic dictionary definition of context as *conditions or circumstances which affect some thing* [WWM75]. Since we focus on recommender systems and the general definition of context is very broad, we restrict it to the domain of recommender systems such as time, place and the company of other people (e.g., for watching movies or dining out).

We briefly discuss the term *context* and detail it in our work in the later chapters. Context information plays a vital role in filtering recommendations or tailoring the right recommendations for the right user. Contextual information such as time, location (GPS coordinates), weather, crowd, weekday/weekend, special events contribute to the quality of recommendations for travel. For example, recommending a park during a rainy day is not a good recommendation inspite of park being highly rated. Location Based Social Networks (LBSNs) are social networks where people provide feedback about places either by checking in to a place, or by

providing star ratings. Well known LBSNs are Foursquare[1], Facebook[2], Google Places[3] where users can rate the places they visit, know about, or add more places which are not available on the networks.

1.3 Proactive Recommendations

A proactive recommender system pushes recommendations to the user when the current situation seems appropriate, withou any explicit user requests [WHBGV11]. For example: a user walking around in the city centre at noon time, might either be looking for a restaurant or a bar (see Figure 1.2). This situation corresponds to contextual situations in recommender systems. This family of recommender systems is an upcoming topic in the field of wearable devices, data mining and recommender systems.

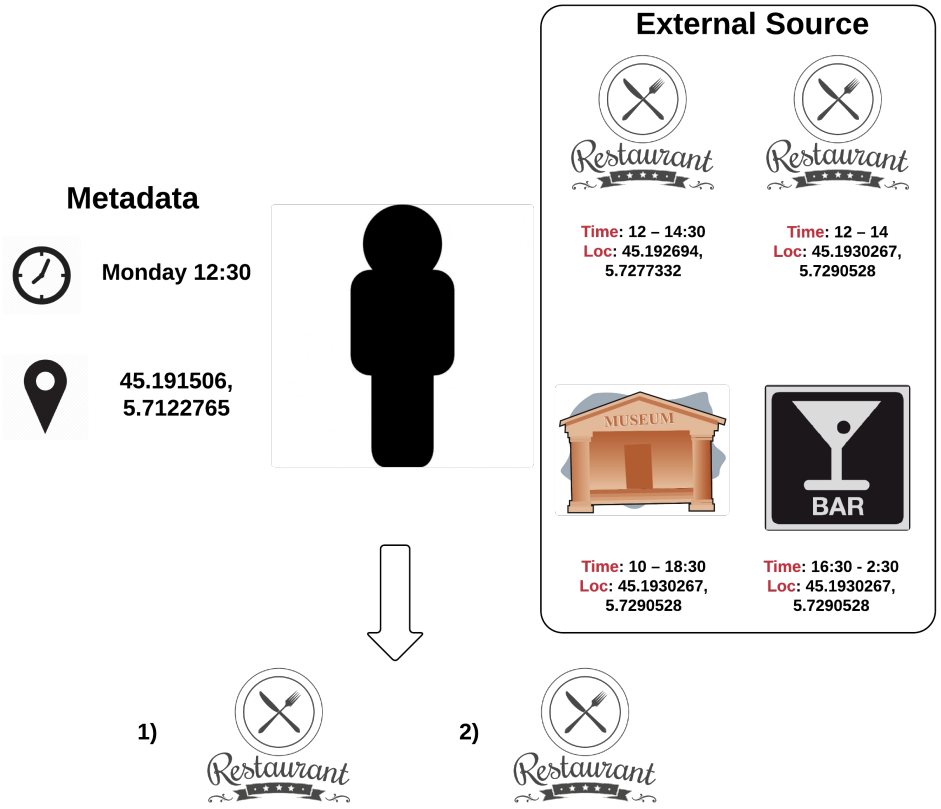

Figure 1.2 – Proactive recommendation example

[1]https://foursquare.com
[2]https://www.facebook.com/help/461075590584469
[3]https://www.google.com/business/

Our work focuses on modeling a contextual recommender system which can take into account, the location, time and multiple feedback as input as provide recommendations at right contextual situations needed for a proactive recommender. Figure 1.2 shows an example of different aspects of a proactive recommender system. A user's device metadata such as time and location are considered in the example. The time in the Figure 1.2 represents an afternoon timeperiod suitable for lunch recommendation. Hence the system ranks the list of places which are open at that time and nearby to the user's vicinity. Nearby places can be calculated using Manhattan distance or any other measure for comparing GPS coordinates [Bla06].

A major difference between proactive and traditional recommender is that the proactive recommenders works on push mode but traditional recommenders works on pull mode, i.e. a user does not request for a recommendation in proactive systems, while in traditional systems the user has to request for recommendations.

Push mode recommendations can be intrusive to users. Users do not like to get recommendations at unsuitable times [DBSR15]. Hence it very important to understand when and where the users should be delivered with recommendations. Context plays a vital role in this aspect of proactive systems for identifying suitable situations for recommendations. Our goal is to model the temporal and spatial information into the recommender model, so that it can be effectively used to provide proactive recommendations.

Knowing the context of the user with the help of device metadata such as time and location facilitates Proactive recommendations [WHBGV11]. Proactivity as defined in Wikipedia[4] - *Proactive behavior involves acting in advance of a future situation, rather than just reacting. It means taking control and making things happen rather than just adjusting to a situation or waiting for something to happen.* In the case of recommendations, the continuous availability of the device metadata will allow the inference of contextual situations [BCP14]. These contextual situations can be hard coded in the system to detect the user's activities. As mentioned before, metadata such as time, date, location can contribute in inferring the contextual situation required for recommendations. Hence on the recommender systems part, the design of the system should take into account such contextual situations before providing recommendations. This is when considering *context* for designing a recommender algorithm becomes important.

1.4 Profile Data vs Feedback Data

Useful data for recommendations can be classified into two categories:

- User profile data - name, gender, age, nationality, etc.
- Feedback data - preferences, ratings, likes, places visited, etc.

A problem occurs when a new user arrives into the system and does not have interacted with the system yet, which results in absence of data about her. This is known as *cold start problem*. Due to this, recommendations cannot be provided to the user until she interacts. Either the user profile information is missing or the system fails to find suitable similar users community for the new user. User profile data are used extensively used in cold starting content-based recommender systems, because of the availability of profile information [NDB07]. Data generated from user's interaction with the system directly or indirectly are called as feedback

[4]https://en.wikipedia.org/wiki/Proactivity

data. Such information portrays an interaction of a user with an item. Our focus is on the feedback data and not the profile data.

Users perform *checkin* or tag a location with their post in social networks, which means that they explicitly share their information of visits to a place to their friends on their network and sometimes even publicly (depending on the their privacy settings). This checkin data consist of the user id, profile link, timestamps, location and more details about the location such as place name, type of place (restaurant, park, gamezone etc.), open hours, location (GPS), complete address etc.

Feedback on LBSNs provided by the users are of two types: implicit feedback and explicit feedback. Explicit feedback are the experience of the user with a POI given by herself. They are in the form of likes (binary acceptance values), star ratings, reviews etc. Implicit feedback are binary values which represents item purchases, movies watched, places visited (yes/no values). We consider checkins as implicit data even though they are explicitly given by users because this information does not provide any likeliness of the place to the user and this data can be obtained implicitly through others means. Inference of activities from the raw GPS data facilitates to know which places were visited by the user, using an external knowledge source [FCRS13]. Through this approach the places visited by the user can be obtained at a given timestamp. This type of feedback are implicitly obtained from the user data hence they are abundant compared to explicit feedback. On the contrary, these do not provide any information about the satisfaction of the user with the item or POI.

The main objective of our work is to consider both implicit feedback and explicit feedback from the users about the POIs and represent it in a suitable way and use this information for providing contextual recommendations which are suitable for that context, which is time and location aware.

The intuition behind using location and time as context for recommendations is that the proactive recommender systems are one of the upcoming class of systems which work also on the basis of these factors. To support such system, the context has to be analyzed and modeled suitably. Building a new algorithm for a new system or a new domain is not productive, instead one can focus on modeling the data in suitable manner for the existing state-of-the-art algorithms.

Definitions which we use in the rest of this document are as followed:

- **Recommender model/system**: system which processes the input data along with external knowledge source to output recommended items
- **Recommendations**: ranked list of items.
- **Context**: location (distance, nearby places) and time (timestamps, timeframes, date).
- **Tensor**: N dimensional matrix like structure (3 Dimensional tensor specifically)

1.5 Problem Statement

As mentioned earlier, several approaches are used to provide recommendations of places for users based on the data collected from the user and other similar users.

Explicit data can be used to determine what the user likes based on ratings, likes provided by the user. Implicit feedback can be used to determine various aspects such as crowd density,

places visited, user's presence in different areas at different time slices etc. Although this information cannot depict the user's likeliness to visit a place, or attributes of the place that he liked,it depicts some of the features which are useful for recommendations. Implicit feedback can be used to provide recommendations based on other user's presence in the targeted POIs as discussed in [YSQ+15].

Existing systems use explicit and implicit feedback individually to provide recommendations.
The problems encountered when implicit feedback is individually considered:

- Less crowded places (crowd presence calculated using checkins) are not recommended to the user.
- Places where a user **U**'s friends have never visited before [YSQ+15], are not recommended to the user **U** (on an assumption that user prefers the places where his friends have been before).

The problems encountered when explicit feedback are individually considered:

- Low rated places (aggregated ratings of a place signifying bad reviews) are not recommended.
- Places without ratings, are not recommended.

1.6 Structure of the report

This document is structured as following: Chapter 2 discusses about work related to our study, then while Chapter 3 details our approach to the problem described above, Chapter 4 shows the experiments we have done to assess our approach. Eventually, Chapter 5 concludes and sketches some further work.

— 2 —
Related Work

In general, recommender systems are tools and techniques which provide users with suggestions for items a user may wish to utilize [RRS11]. These items can be songs, shoes, shirts, movies, places to visit (POIs), events, computers, mobile phones etc. These suggestions, facilitates the service providers to boost their sales by targeting the right crowd with the right set of products. There are various reasons for service providers to use a recommender system for their services. Some of them are: increase the number of items sold, sell more diverse items, increase user satisfaction, increase user fidelity, better understand what user wants [RRS11].

Recommender systems are used in various domains, and some of the major service providers include *Netflix* (movie/tv show streaming service provider), *Amazon* (shopping website), *Spotify* (music streaming service provider), *Quora* (question-and-answer website where questions are asked, answered), *Swarm* (an application based on Foursquare services), and so on. We discuss below some of the notations, techniques, approaches used in recommender systems:

Users

Users are the stakeholders who are involved in providing feedback and receiving recommendations, their profile contain additional information about a user, consisting of attributes like gender, age, address, nationality, choices, preferences etc. Some of these attributes are explicitly provided by the user and some are implicitly assumed, some of the implicit assumptions are the location of the user based on her IP address, her clicks on certain items, products purchased in the past etc.

Items

Items are the products which are provided by the service providers to the user. They are domain dependent. Sometimes these items can also be other persons, groups based on the purpose of recommendations. In our work, we use **POIs** as items.

Feedback

Feedback are the results of Interaction of user with the system. We discuss in detail about the feedback in section 3.1. Generally, feedback are ratings provided by the users for items, which are numerical values in ranges. For example, 0-5 stars for a movie, binary rating for liking/unliking a product etc.

2.1 Types of Recommender Systems

Recommender systems generally are classified into three types.

- Content-based recommenders.
- Collaborative filtering based recommenders.
- Hybrid recommenders.

Content based Recommenders

Content-based recommenders work on the basis of a simple assumption that, if a user likes something in the past, he might like similar items in the future. Similarity scores are calculated with items rated by the user and items which are not rated by the user.

Content-based recommenders are used for personalizing recommendations for users. When a new user enters the system, she would not have interacted with the system hence the data required to provide recommendations will be lacking, this is a drawback of content based systems.

Diversity problem is another problem where a user always rates similar kind of items (same genre movies) which results in recommendations of only that category of items.

Collaborative filtering Recommenders

Collaborative filtering techniques are used to provide recommendations based on similarities between the user U and other users or similarities between the items. Similarities can be between the items which user U and other users interacted with, or the similarities can be between the users.

One of the major problems which collaborative filtering systems suffers from, is the cold-start problem because the system would not understand the similar users unless the user has performed some interactions [NDB07].

Hybrid Recommenders

Hybrid recommenders are a special type of recommenders which are often used in commercial scenarios to avoid the cold start problem, diversity problem [BCR14].

Recommendations are domain dependent and defining context is dependent on the recommendation environment [Dey01, AM06]. As the variety of items in different domains differ, the word *context* also changes its meaning as we discussed earlier.

In collaborative filtering approaches two primary methods are:

1. Neighborhood methods are centered on computing the relationships between items or between users. The item oriented approach evaluates a user's preference for an item based on ratings of neighboring items by the same user. A product's neighbors are other products that tend to get similar ratings when rated by the same user [KBV09].

2. Latent factor models are an alternative approach that tries to predict the ratings by characterizing both items and users with some number of inferred factors from the rating patterns, say 20 to 100 factors. In a sense, such factors comprise a computerized alternative to the human made factors or the profile details as used in content based systems.

For instance, a user can be described by the user profile attributes such as gender, age etc. which are human made and observable factors, whereas relationships between items and users cannot be specified, as they vary according to the data and are numerical patterns, hence they are inferred from the data values.

2.2 Context in Travel Recommendations

The contextual factors or situational factors which help in describing a situation, will be focused on aspects which a traveler would be interested in. We use only location and time as the two contextual variables to consider nearby places and timely recommendations, similar to works discussed in [RRS11, YSQ+15].

For instance, a user who is looking for a restaurant nearby her vicinity would like to know the restaurants nearby to her and which are open. Providing the best restaurant recommendation to her which is 200 miles far from him will not be suitable. Hence location and temporal factors contribute for suitable recommendations [KGF16, FKCC+15].

2.3 Matrix and Tensor models

Some of the successful and widely used latent factor models for recommendations include matrix factorization approach. Matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns [KBV09].

Recommender systems represent the data in the form of matrices with one dimension as the user and the other as the item, the values are the rating provided by the user. Matrix factorization approaches such as non-negative matrix factorization [LS01], probabilistic matrix factorization [SM11], bayesian probabilistic matrix factorization [SM08] are widely used in the field of machine learning, data mining, recommendations etc.

Matrix Factorization

Basic matrix factorization model as shown in the Figure 2.1:

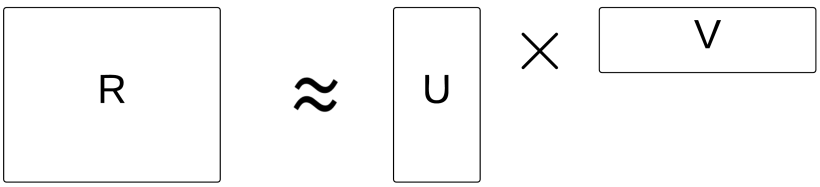

Figure 2.1 – Matrix R factorized into U and V.

A matrix R consists of ratings provided by users U on the items V. Items are considered as V because items in travel domain are venues (or POIs). The goal of matrix factorization

is to find two matrices **U** (a $|U| \times K$ matrix) and **V** (a $|V| \times K$ matrix) such that their product approximates **R**, and K represents the rank of the decomposed matrices.

$$R \approx U^T \times V = \hat{R} \tag{2.1}$$

In this way, each row of **U** would represent the strength of the associations between a user and the features. Similarly, each row of **V** would represent the strength of the associations between an item and the features. To get the prediction of a rating of an item v_j by u_i, we can calculate the dot product of the two vectors corresponding to u_i and v_j:

$$\hat{r}_{ij} = u_i^T v_j = \sum_{k=1}^{k} u_{ik} v_{kj} \tag{2.2}$$

Once the ratings are estimated, it is necessary to calculate the difference between their product to the real values in **R**, and then try to minimize this difference iteratively. Such a method is called gradient descent, which tries to optimize the local minima.

Squared error is the error between estimated ratings and actual ratings, raised to the power of 2, is calculated by the following equation:

$$e_{ij}^2 = (r_{ij} - \sum_{k=1}^{K} u_{ik} v_{kj})^2 + \frac{\alpha}{2} \sum_{k=1}^{K} (||U||^2 + ||V||^2) \tag{2.3}$$

The parameter α is used to control the magnitudes of the user-feature and item-feature vectors such that P and Q would give a good approximation of **R**.

Matrix factorization models are widely used in recommender systems to factorize large matrices into smaller matrices and find similar users or similar items based on certain requirements of the domain.

Tensor Factorization

Tensors could be seen as a vector in an n-dimensional space[1]. The main idea behind the use of tensors is that we can take advantage of the same principles behind matrix factorization to deal with N-dimensional information [KABO10]. Traditional recommendation systems use matrix representations to store data and perform computations on it. Those approaches using matrices are successfully used in many domains and applications. Matrix Factorization is one of the most used approaches for collaborative filtering but the model is not flexible enough to add contextual dimensions in a straightforward manner as the matrix can only hold 2 dimensional information [KABO10]. Tensors can hold more variables in different axis which allows to perform easier computational mechanism when more contextual variables are introduced. Several works are reported on using tensors for recommendations (see for instance [XCH$^+$10, KABO10, YZZY15, YSQ$^+$15]).

The definition for basic tensor factorization model for POI recommendations using location and time as follows [YSQ$^+$15]:

Tensor $\mathcal{R}$ represented in the figure 2.2 is constructed using m users $\{u_i\}_{i=1}^{m}$ on n venues $\{v_j\}_{j=1}^{n}$ at q timestamps $\{t_k\}_{k=1}^{q}$. The tensor $\mathcal{R} \in \mathbb{R}^{m \times n \times q}$ is a third order tensor, where each $\mathcal{R}_{i,j,k}$ contains the feedback values.

[1] https://en.wikipedia.org/wiki/Tensor

The tensor $\mathcal{R}$ can be estimated from the sparse values using CP decomposition D component of rank 1 tensors [KB09]. The preference of a user or the number of visits of user u_i on location l_j in time frame t_k can be approximated as: $\mathcal{R} \approx \sum_{d=1}^{D} u_d \circ v_d \circ t_d$ where $\hat{\mathcal{R}}$ denotes the predicted approximation of $\mathcal{R}$; $u_d \in \mathbb{R}^m$, $v_d \in \mathbb{R}^n$, $t_d \in \mathbb{R}^q$. The aim is to find the decomposition of $\hat{\mathcal{R}}$ that best approximates the original tensor $\mathcal{R}$ to achieve better results (see Figure 3.2). Optimization problem by minimizing the square loss is as follows:

$$L(U,V,T) = \frac{1}{2} \min_{U,V,T} \| \mathcal{R} - \hat{\mathcal{R}} \|_F^2 \tag{2.4}$$

$$= \frac{1}{2} \min_{U,V,T} \| R - \sum_{d}^{D} u_d \circ v_d \circ t_d \|_F^2 \tag{2.5}$$

Where $U = [u_1, ..., u_D] \in \mathbb{R}^{m \times D}$, $V = [v_1, ..., v_D] \in \mathbb{R}^{n \times D}$, and $T = [t_1, ..., t_D] \in \mathbb{R}^{q \times D}$ are factor matrices.

To avoid overfitting, the regularization terms associated with U,V,and T are introduced into equation 2.5 as:

$$\min_{U,V,T} \frac{1}{2} \| R - \hat{R} \|_F^2 + \frac{\lambda}{2} (\| U \|_F^2 + \| V \|_F^2 + \| T \|_F^2) \tag{2.6}$$

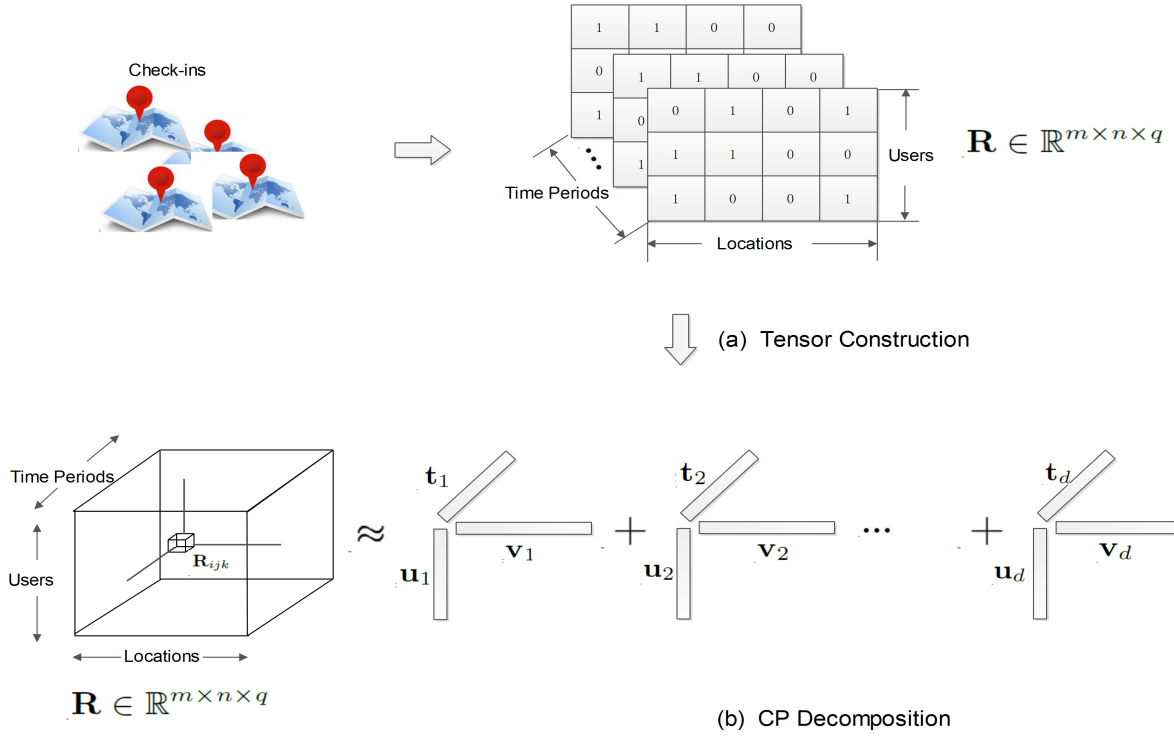

Figure 2.2 – Tensor representation (figure taken from [YSQ+15])

Several approaches use tensors for computing missing values or hidden relationships within the data. But there are certain problems such increased sparsity value compared to matrix approaches because of the increase in dimensional. The work reported in [KCL15] shows the tensor factorization via matrix factorization. Authors state that the numerical methods for tensor factorization have not reached the level of maturity of matrix factorization methods. Hence they propose a new algorithm for CP tensor decomposition that uses random projections to reduce the problem to simultaneous matrix diagonalization.

Feedback data in recommendations

Data in a matrix or a tensor are numerical values which can be of range (0 - 5 or 1 - 10) in the case of ratings or binary (Y/N) in the case of implicit feedback. Implicit feedback might also be a counter value which increases the count, everytime a user performs an interaction. The idea of complementing explicit with implicit feedback was first proposed in [BK07b], where the explicit feedback were the ratings for the movies and implicit feedback were the movies which were rated by the users. A factorized neighborhood model was used to combine both the forms of feedback, which is an extension of traditional nearest item-based model.

Another method which unifies the explicit and implicit feedback for collaborative filtering was discussed in [LXZY10], where authors normalize both forms of feedback into one scale of values and propose a co-rating model as shown in equation 2.7:

$$arg\min_{U,V} L_E(U,V) + \eta.L_I(U,V) + \lambda.\mathbb{R}(U,V) \tag{2.7}$$

In Equation 2.7, square losses due to explicit and implicit feedbacks are added together and an additional parameter η is introduced to weight the relative importance of two feedbacks.

The above mentioned work has been focused on collaborative filtering approaches. The work discussed in [LC16] is one of the first approaches to consider the use of both type of feedbacks into recommender model for personalized ranking. Authors developed a new ranking algorithm (MERR_SVD++) based on the xCLiMF [SKB+13] and SVD++ [Kor08] which exploits both the explicit and implicit feedback simultaneously and optimize the well known evaluation metric, so called Expected Reciprocal Rank (ERR). These approaches use a matrix to represent data which uses a 2 dimensional representation.

Datasets

Most of the research works use datasets released by the following content providers which are anonymous and free for research use: *Netflix*[2] for the Netflix challenge[3], *Kaggle*[4], *Stanford Large Network Dataset Collection*[5], *UC Irvine Machine Learning repository*[6] etc. The Netflix Prize was an open competition for the best collaborative filtering algorithm to predict user ratings for films, based on previous ratings without any other information about the users or

[2]https://www.netflix.com

[3]https://en.wikipedia.org/wiki/Netflix_Prize

[4]https://www.kaggle.com/datasets

[5]https://snap.stanford.edu/index.html

[6]http://archive.ics.uci.edu/ml/index.html

films, i.e. without the users or the films being identified except by numbers assigned for the contest. Netflix provided a training data set of 100,480,507 ratings that 480,189 users gave to 17,770 movies. Each training rating is a quadruplet of the form (user, movie, date of grade, grade): Information borrowed from Wikipedia article[7].

2.4 Synthesis

2.4.1 Context-awareness

Context has been studied extensively in various disciplines such as smart homes, smart cities, information retrieval, ubiquitous computing etc. Importance of using context in recommender systems has been discussed a lot of times, we consider context to be an important factor in the recommender model.

In travel recommendations, context can be several things such as time, location, weather, friends (social), weekday/weekend, seasons etc. As we are limited to certain available datasets, we consider Time and Location to be our contextual factors.

2.4.2 Tensor Model

As there is a clear advantage of using tensors over matrices inorder to consider higher dimensional data, we decided to use tensors to represent the data (feedback). Specifically we use a 3 way tensor with User, Time and Location representing each axis. And the tensor data can be explicit, implicit or combined feedback which we discuss in detail in Section 3.1.

[7] https://en.wikipedia.org/wiki/Netflix_Prize

— 3 —
The ExIm Model

3.1 Proposed Approach

Our objective is to model data useful for proactive recommender systems, hence we propose to use the feedback data, consider location and time variables into a tensor representation for feedback values. The goal is to leverage the existing state-of-the-art techniques and algorithms for providing recommendations, while representing heterogeneous feedback data in existing recommender models. Working of the proposed recommender system is briefly explained below and represented by the Figure 3.1.

A recommender system has three main components:

1. Input data - explicit, implicit, external
2. Recommender model - similarities and scoring methods
3. Recommended items (Output) - ranked list of items

Input data can be further divided into three categories as listed above: 1) explicit data being the ratings provided by the users, 2) implicit data being the checkin data of the users in a POI, 3) external data being the data about the POIs which are namely the name of the place, type, address, location (coordinates), specialties, timings.

Detailed discussion about the data collection and various types of issues are discussed below.

3.2 Similarity between Users and Items

Delivery of recommendations for a user depends on what similar users have liked or visited in the past. Hence calculation of similar users is necessary to deliver recommendations. We can use Jaccard similarity coefficient as a measure to calculate to similar POIs based on their feedback values.

The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets[1]. Jaccard similarity is

[1] https://en.wikipedia.org/wiki/Jaccard_index

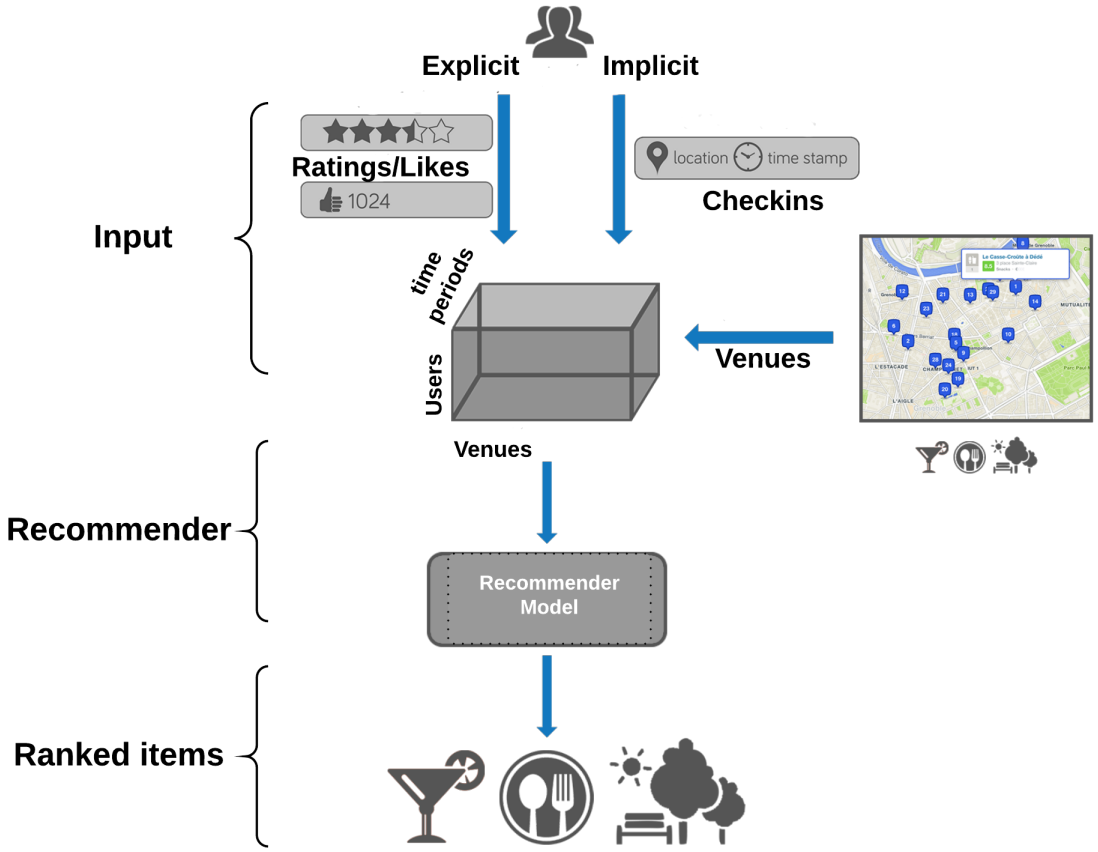

Figure 3.1 – POI recommender with dual feedback as input.

used to measure the similarity between two set of elements. In the context of recommendation, the Jaccard similarity is used to compute similarity between two items as shown in equation 3.1.

$$J(a,b) = \frac{|U_a \cap U_b|}{|U_a \cup U_b|} = \frac{|U_a \cap U_b|}{|U_a| + |U_b| - |U_a \cap U_b|} \tag{3.1}$$

where U_a is the set of users who rated item a and U_b is the set of users who rated item b. Jaccard similarity coefficient gives the measure of similarity between these two sets of users. Jaccard is a good choice when one only has implicit feedbacks of items (e.g., people rated them or not). Based on similar unseen items for a user U, new recommendations can be delivered.

3.3 Data Modeling

The data consist of users U who have performed checkins at venues V and timestamps T.

Traditional recommender systems use matrix representation of data which can model the user and item relationship, using ratings/likes or implicit data such as purchases histories. In

order to consider more variables or more features in this user and item relationship, a higher dimensional approach is required.

For instance, considering time as an extra variable for a retail store sale recommendation can utilize temporal entities in the purchase history to provide time-aware recommendations, where time is represented using seasons [XCH+10].

For travel and tourism purpose considering time/temporal entities into the recommender model facilitates time-awareness for the recommendations. Some works use a tensor representation for this purpose [XCH+10, KABO10, BLR11] . In order to model the context into the recommender model, we use a tensor representation of the data.

Figure 3.2 represents the Tensor representation of the data where users U, venues or POIs V and timestamps T are modeled as 3 axis of a three dimensional tensor. The values of the tensor are of three different types. Frequency of user visits, ratings from the places and both combined by normalization.

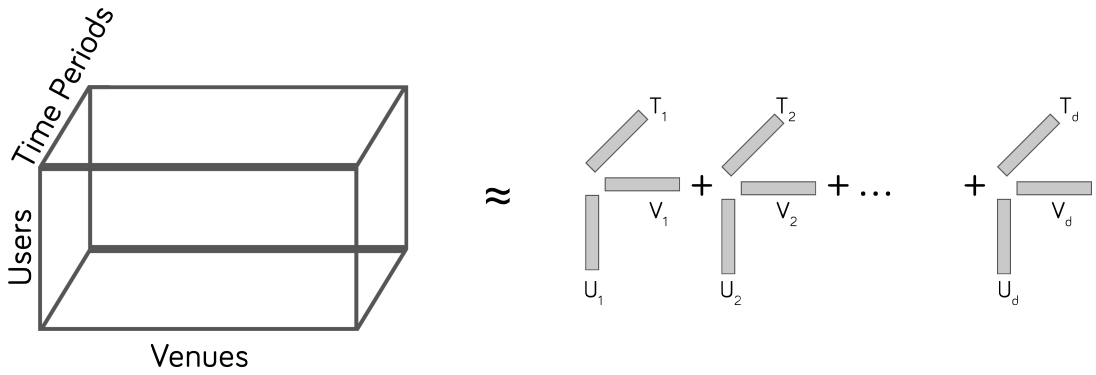

Figure 3.2 – Tensor representation of the data.

3.3.1 Explicit feedback

3.3.2 Dataset

Explicit data is the data explicitly given by the user, such as ratings, likes etc. As shown in Figures 3.3 and 3.4, explicit data is data available on the web which determines how well a place or an object was liked by the users. This data is often trustworthy for other users. Although type of data can be easily accessible on the web, the quantity of this data is very less. This issue is called sparsity problem in recommender systems. The users interacting with the system will not rate all the items in their vicinity, and also not all the items will be seen the

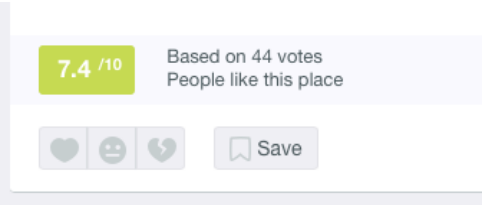

Figure 3.3 – Ratings on Foursquare.

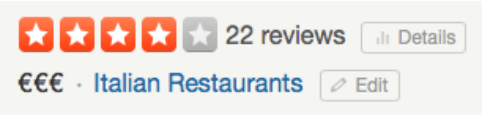

Figure 3.4 – Ratings on Yelp.

user, which gives rise to missing data, which is a problem for algorithms built on assumption that the data is complete. In travel domain, users do not rate all the POIs which visited by them, which makes the dataset sparse. The work reported in [MZ09] shows how the users rate while asked to rate items in random.

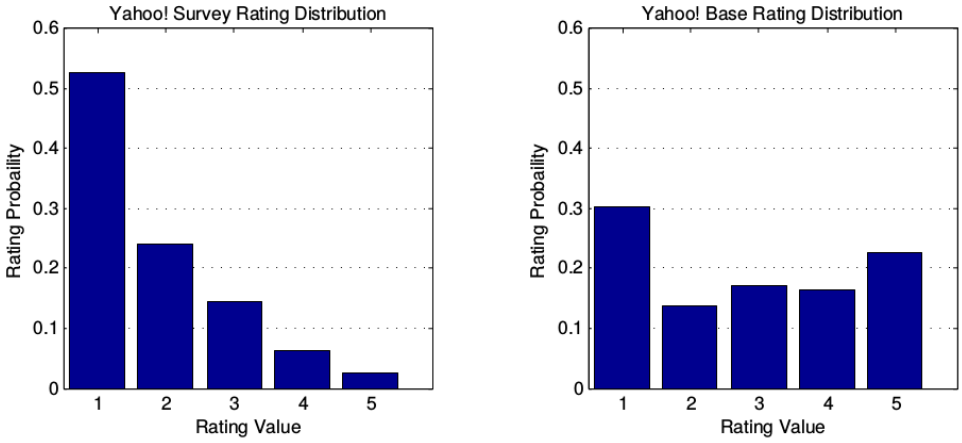

Figure 3.5 – (Left) Limited list of item ratings; (Right) Free list of item ratings.

Figure 3.5 represents the results of two cases of user responses. Users when provided with a limited random set of items and asked to rate, provided low ratings for some items (Fig.3.5 Left). But when users were provided with a free list of items and asked to rate random items which they chose to rate, the ratings obtained were better in quality (Diverse ratings) (Fig.3.5 Right) [MZ09].

This study reflects that the users tend to rate better when they are not forced to rate, but given a freedom of choice to rate the items, the rating quality gets better. This makes it harder to obtain such quality feedback in realtime.

3.3.3 Implicit feedback

Implicit data is indirectly obtained from a user's interaction with a system. Such as mouseclicks, purchases history, visited webpages, visited POIs, etc. Implicit data for travel/tourism is the location of the user and the timestamp. Implicit data is generally available in abundance when compared to the explicit data as the explicit data is often very sparse [BK07a]. Implicit data can be obtained in many ways, in our work we consider the checkin from Foursquare [YZZY15] as implicit data because this data even though its explicitly given by the user, it still does not express the likeliness or directly depict the reviews of the visit. Also, many other social networks such as Facebook[2] and Twitter[3] allow tagging of location with the status and tweets respectively which represents a user's presence at that instance. Hence we consider it as Implicit data.

[2]https://www.facebook.com
[3]https://www.twitter.com

The use of LBSN (Location Based Social Networks) might be fading away in the recent years as people are getting used to new networks and new types of data is being generated. There was a study about "Context Collapse" in Facebook, where people's information or expectations from one context invade or encroach upon another[4]. Users share lots of content on their Facebook profiles and tweet a lot about things happening around the world on Twitter but people are sharing less about themselves such as sharing about their whereabouts, their trips, their status, their checkins etc. Hence obtaining this sort of implicit data using networks such as Foursquare, Facebook might become obsolete in the upcoming years as there are privacy concerns about this and users feel less secure.

We acknowledge that the privacy of the user is ignored in our work, the security of the data is out of scope for this work but we feel that there are needs for secure infrastructures, maintaining anonymity, keeping users in the loop when procuring such data.

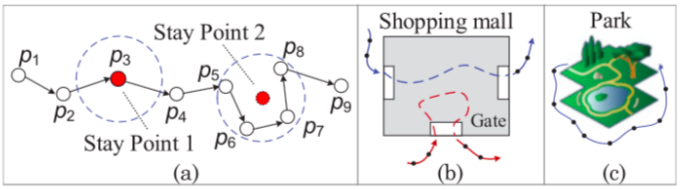

Figure 3.6 – Stay point calculation in a GPS track

There are several uses from the implicit data of the user. We studied several alternatives for obtaining this type of data apart from the Foursquare dataset which we use. One of the approaches was analysing GPS Tracks or GPS trajectories of a user in a given time frame and extracting the places he visited, as discussed in [FCRS13]. The authors of this work extracted the places from GPS tracks using stay points. A stay point as shown in figure 3.6 is a point in the GPS track where a user has stopped for more than a threshold value of time. This threshold value can range from 20-40 mins for the POI in travel. Knowing the stay points, the visited locations can be determined using Google Maps[5] data or Foursquare API data to determine the places around that stop point and assume the closest one as the place visited by the user. In this work [FCRS13], authors try to infer the activities carried out by the user, by knowing the place visited by the user, by the method mentioned above.

The work discussed in [ZX10] takes into account a user's travel experience and the sequentiality that locations have been visited, then they learn the location correlation from a large number of user-generated GPS trajectories.Using this location correlation, a personalized location recommendation system was built and evaluated on a real-world GPS dataset collected by 112 users over a period of 1.5 years.

[4]https://www.theguardian.com/technology/2016/apr/19/facebook-users-sharing-less-personal-data-zuckerberg

[5]https://www.google.com/maps

3.3.4 Contextual Factors

Contextual factors such as time and location helps in providing relevant recommendations to a user who is travelling. Though we are not building a recommendation system in this work, the recommender model which is built on considering the context as input helps to build a recommendation system easily.

Building a proactive recommendation system needs a context aware recommender model which is able to handle the contextual situations, which a proactive recommendation system would work on. Hence understanding the context is the task of the recommendation system. But building the recommender model in order to consider such contextual factors is the challenge which our is focussing on.

Explicit feedbacks which are represented in a range of numbers and implicit feedbacks which are binary values in our case are to be unified into one value before proceeding further. The reason why this has to be done is because most of the well known algorithms for recommendations rely on one sort of feedback. For instance [BLR11, XCH+10] use ratings as the primary data to provide recommendations. But many implicit data are ignored while these methods are utilized.

Users not only depend on rating scores to decide on their choice, there are other contextual factors which might influence user's choice.

Hence there is a need to adapt the heterogeneous data into existing algorithms to utilize the potential of the state of the art system to the maximum.

Section 3.4 represents the possibilities of the feedback data considered individually and combined together, using fuzzy logic.

Cases such as:

- Low rated place but highly crowded.
- High rated place but low crowded.

can be of interest during special events or specific timeline of a year. This data plays vital role when temporal events are considered into recommender models.

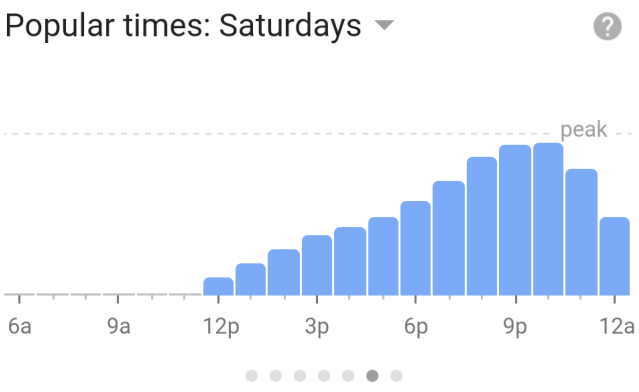

Figure 3.7 – Popular timings for a place shown in Google.

Combining or unifying both the feedback facilitates to provide context aware recommendations with respect to both the likeliness of the places as well as contextual factors. Two cases where combination of feedback facilitates recommendations: 1) low rated but crowded places. 2) less crowded but highly rated places.

3.4 ExIm Tensor model

ExIm stands for Explicit and Implicit data. We consider explicit feedback as x_{ui} and implicit feedback as y_{ui}. There are two complications in considering both the feedback together.

- x_{ui} are ratings, which are often in a range of 1-10 or 0-5, where as y_{ui} are the user's presence in a venue, which is either -1 or +1 or binary values.
- The range of x_{ui} and y_{ui} are highly variant with each other hence they cannot be unified or combined directly.

To address these problems, [LXZY10] normalize the original values x_{ui} and y_{ui} to a common scale ranging from 0 to 1 and assign different importance to explicit and implicit feedback using η as a parameter. Normalizing both the feedback to binary values leads to loss of valuable information obtained by the ratings. Normalizing the values remains suitable for binary cases of explicit data (ratings) such as *Like* and *Unlike* values but not when the values are ternary or n-ary or in a range. To model the data of crowd presence and ratings together, there is need for common representation and binary representation will not hold good because of the loss of information.

Hence we propose a model based on Fuzzy logic [Tan97, Men95] for combining both feedback sets. We list the steps of fuzzy logic process for our work and describe each in detail:

1. Define the linguistic variables and terms.
2. Construct the membership functions.
3. Construct the fuzzy matrix based on predefined rules.
4. Use it for filling tensor values.

Linguistic variables are the input or output variables of the system whose values are words or sentences from a natural language, instead of numerical values. We define two sets of input linguistic variables for explicit and implicit data.

Implicit data in travel domain are the crowd presence calculated using the checkin data. Checkin data is raw data which has to be processed to understand the crowd presence. Hence we aggregate the checkin data over a timeperiod and compare the maximum and minimum count of checkins as shown in Figure 3.7, at a time interval such as weekday or weekend, and represent this raw checkin data in terms of three meaningful variable as shown in equation 3.2 by substituting minimum value to *empty*, maximum value to *crowded* and average values to *moderate*.

$$T(Im) = \{empty, moderate, crowded\} \tag{3.2}$$

Explicit data is often in the range of 1 - 10 or 0 - 5, to have a generic representation of these values, we consider 5 values to represent the explicit feedback data. If the range of ratings contains 10 values, it can be reduced to 5 values for this representation to be suitable for the data.

$$T(Ex) = \{worst, bad, average, good, very\,good\} \tag{3.3}$$

Membership functions for $T(Ex)$ and $T(Im)$ are plotted in figures 3.8 and 3.9 respectively. We have considered the scale of ratings to be 0 - 5 and scale of crowd presence from 0 - 3. As mentioned earlier, the scale of crowd presence is calculated by processing the raw data into three meaningful variables for fuzzy logic.

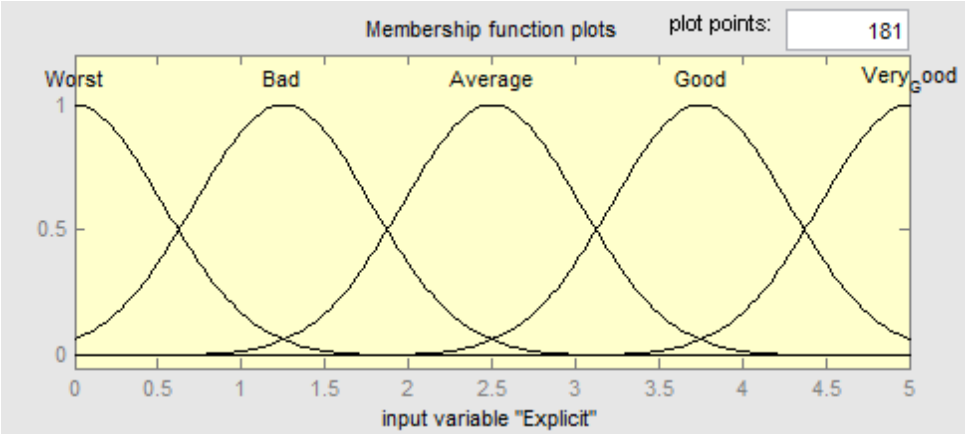

Figure 3.8 – Membership function for Varible $T(Ex)$

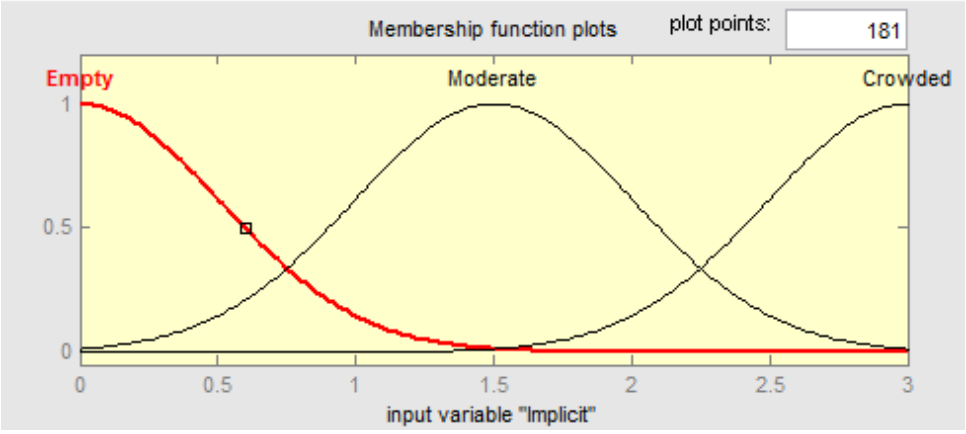

Figure 3.9 – Membership function for Varible $T(Im)$

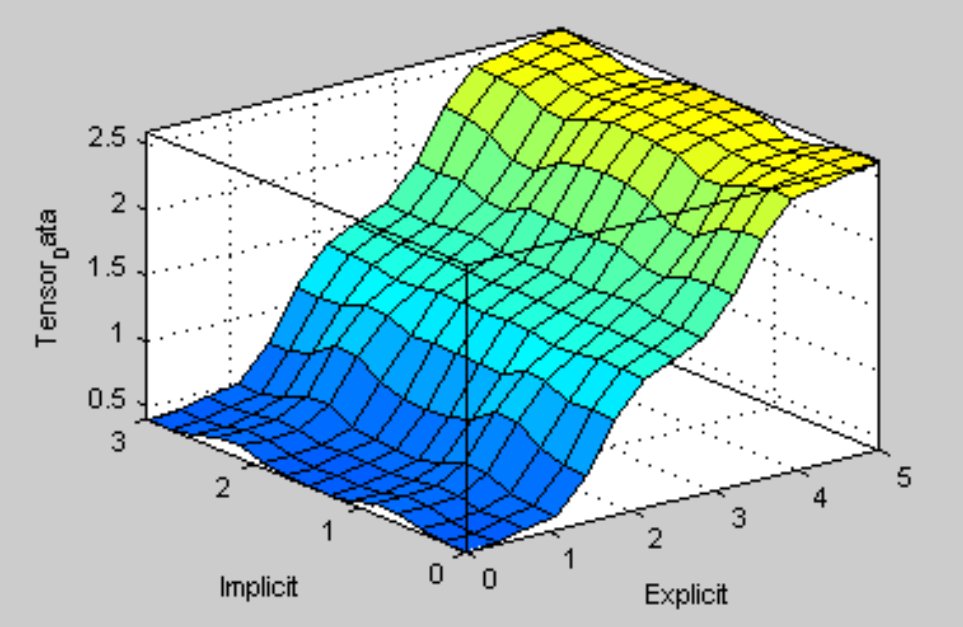

Figure 3.10 – Data of the tensor T.

Implicit/Explicit	Worse	Bad	Average	Good	Very Good
Empty	3	3	2	1	1
Moderate	3	3	2	1	1
Crowded	3	3	2	1	1

Table 3.1 – Fuzzy matrix for representing processed implicit and explicit data.

The tensor contains the data based on the values in fuzzy matrix shown in 3.1 and graphical representation of possible tensor values for different explicit and implicit data are shown in the figure 3.10. A subset of fuzzy rules which were used in generating the fuzzy matrix are shown below:

- If (Explicit is Very_Good) and (Implicit is Crowded) then (Tensor_Data is 1)
- If (Explicit is Average) and (Implicit is Moderate) then (Tensor_Data is 2)
- If (Explicit is Bad) and (Implicit is Empty) then (Tensor_Data is 3)

Complete table with the all the fuzzy rules is given in the Appendix.

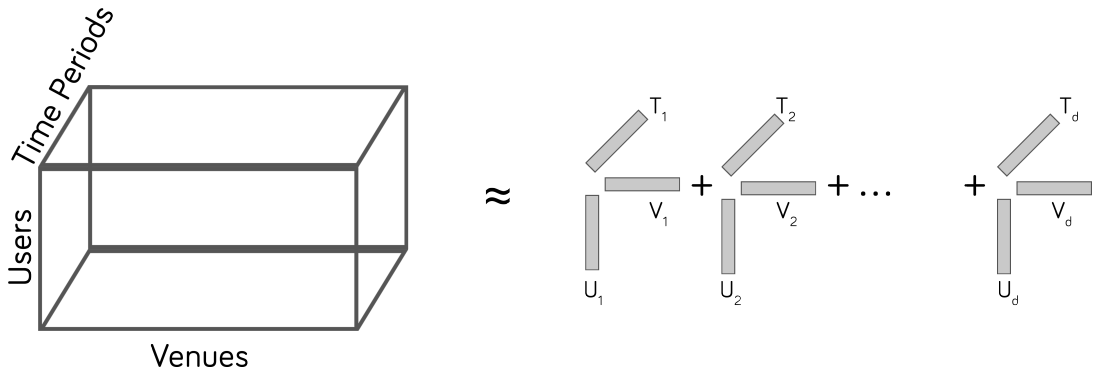

Figure 3.11 – Tensor decomposition.

Tensor R represented in the figure 3.11 is constructed using m users $\{u_i\}_{i=1}^{m}$ on n venues/location $\{v_j\}_{j=1}^{n}$ at q timeframes $\{t_k\}_{k=1}^{q}$. The tensor $\mathcal{R} \in \mathbb{R}^{m \times n \times q}$ is a third order tensor, where each $\mathcal{R}_{i,j,k}$ contains the feedback values generated by fuzzy logic.

— 4 —
Experiments

The objective of a good recommender system is to have maximum acceptance rate for the recommendations delivered. Hence it is important to evaluate our proposed ExIm tensor model by recommending some POIs and comparing them based on some performance metrics.

Evaluations in recommender systems are carried out for two different purposes:

1. Rating prediction.
2. Item recommendation accuracy.

Our work is based on POI recommendations to traveling users, it belongs to the class of *item recommendations*.

Item recommendations are ranked list of items delivered to users, and is detailed in Section 3.1. This ranked list of items can be evaluated in real time by asking the users to interact with the system and by observing their feedback. But in an offline scenario, we lack the real user interactions with the system. It is expensive in terms of time and money to conduct evaluations with real time users.

There are three kinds of evaluations:

1. Offline evaluation: offline evaluations are suitable and cheaper solutions because real users are not asked to evaluate the system. *POI recommendations* can be evaluated offline by comparing the output ranked list of POIs with the actual POIs visited by the user, an error value is incremented if the place visited, is not on top 1 position of the ranked list. If the place visited is on top 1 position, then the error value is not incremented. Mean error value is calculated on the entire dataset to obtain the performance score. Only top 1 can be evaluated in an offline case for POIs , as we cannot determine the user's response in acceptance of any other ranked item in the list, as we only know one place visited by the user at that instance [YSQ+15]. Hence this is one of the limitations of offline evaluations.
2. Online evaluation: A recommender algorithm is built and deployed on a real-time system such as an e-commerce site, movie streaming site, etc. and based on user behavior, involvement, increase in sales, increase in site traffic, the performance of the recommender algorithm (system) is measured. This is the most expensive approach, as bad algorithms might cause loss for the service provider.

3. User studies: A system or a prototype is built and users are asked to use the system by informing the users about the evaluation being carried out. This is comparatively cheaper to online approach but this does not always guarantee the best results as the users being evaluated might be biased in some aspects. Hence proper care has to be taken to involve diverse and unbiased class of users.

In our work, the suitable evaluation is offline evaluation because of the ease in conducting it in short timeperiod.

4.1 Experiment settings

In offline evaluations, a dataset of places visited can be evaluated by splitting it into two parts namely *train* and *test* datasets. It is called as A/B testing. A dataset is split into two parts A and B, A represents training part and B represents testing part. This splitting can be done in random, but an important consideration is that the complete set of users should be present in both the train and testing data, but the items interacted by the users can be split. In our work, we take into account all the users into the training and testing data, but POIs visited by the user (which represents items) are split into two parts. Depending on the experiment settings, this can be split in different percentages. i.e. 50% train and 50% test or 70% train and 30% test, etc.

To do an offline evaluation on our dataset, we had to split Foursquare dataset, which is explained in Section 4.2.

Calculation the precision and recall values on our data using Equation 4.1 and 4.2:
Four possible outcomes for the recommended items and test data items, as shown in table 4.1 [SG11].

-	Recommended	Not Recommended
Visited	True Positive (tp)	False Negative (fn)
Not visited	False Positive (fp)	True Negative (tn)

$$Precision = \frac{\#tp}{\#tp + \#fp} \tag{4.1}$$

$$Recall = \frac{\#tp}{\#tp + \#fn} \tag{4.2}$$

 # represents the number of elements.

Other evaluation measures for error rate calculation of rating predictions are:

- Root Mean Square Error (RMSE)
- Mean Absolute Error (MAE)
- Non Discounted Cumulative Gain (NDCG) etc.

For item recommendations, evaluation measures which are used in Information Retrieval (IR) are used. Precision and Recall values are suitable for evaluating Top N items. *N* represents the number of elements in the ranked list to be considered for performing evaluation. Top *N* values are computed for different number of *N* to compare *Precision N* and *Recall N*.

4.2 Datasets

Data required for our ExIm model has to contain *explicit* and *implicit* data of the same set of users, so the evaluation would be consistent.

To evaluate our approach, we assessed several freely available datasets for research purpose. Table 4.2 tabulates the different datasets and their relevance to our work of explicit and implicit data for recommendations.

Datasets	Explicit	Implicit	Remarks
Foursquare	✗	✓	Checkins + aggregated ratings
Yelp	✓	✗	Ratings + aggregated checkins
Brightkite	✗	✓	Checkins only
Gowalla	✗	✓	Checkins only

Table 4.2 represents the availability of data from different service providers. It is evident from the table that we could not obtain a dataset which has both *explicit* and *implicit* data from the same set of users. Yelp and Foursquare although, provide both *explicit* and *implicit* data, they are not from the same set of users. Because Yelp provides checkin data aggregated over numerous checkins of users and Foursquare provides rating data aggregated over numerous ratings and other factors which Foursquare does not disclose. Brightkite and Gowalla datasets belong to LBSNs and have checkin data but does not provide any ratings related to the venues. Hence we could not use them for evaluations.

A brief overview about the dataset we obtained from [YZZY15]:

The Foursquare dataset contains checkins in New York city collected for about 10 months (from 12 April 2012 to 16 February 2013). It contains **227,428** checkins in New York city. Each checkin is associated with its timestamp, its GPS coordinates and its venue categories on the Foursquare platform. This dataset was originally used for studying the spatial-temporal regularity of user activity in LBSNs.

We had to filter the Foursquare dataset with checkins of users who had less than 5 checkins because the evaluation would not be better with less train and train data. We removed places such as schools, workspaces, funeral homes, homes, recycling facilities, car washes, general stores, cemetery, laundries, airports, trains stations etc. as they are not suitable for recommendations.

The dataset collected, refined and later used for experiments contained users, venues and time as shown in the following figures 4.1, 4.2

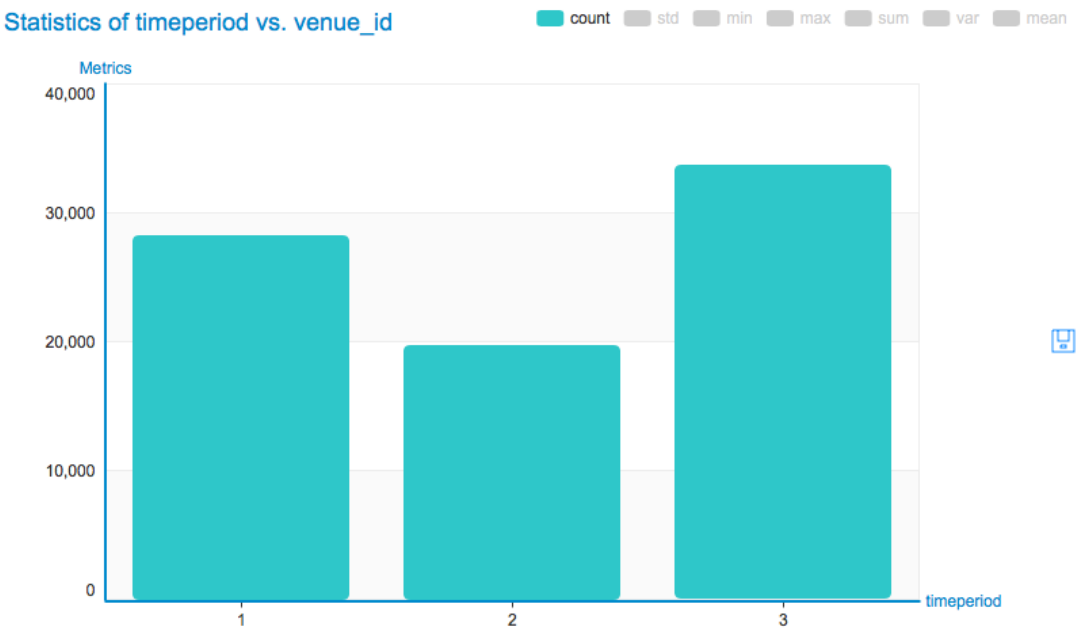

Figure 4.1 – Distribution of Timeperiods vs Venues.

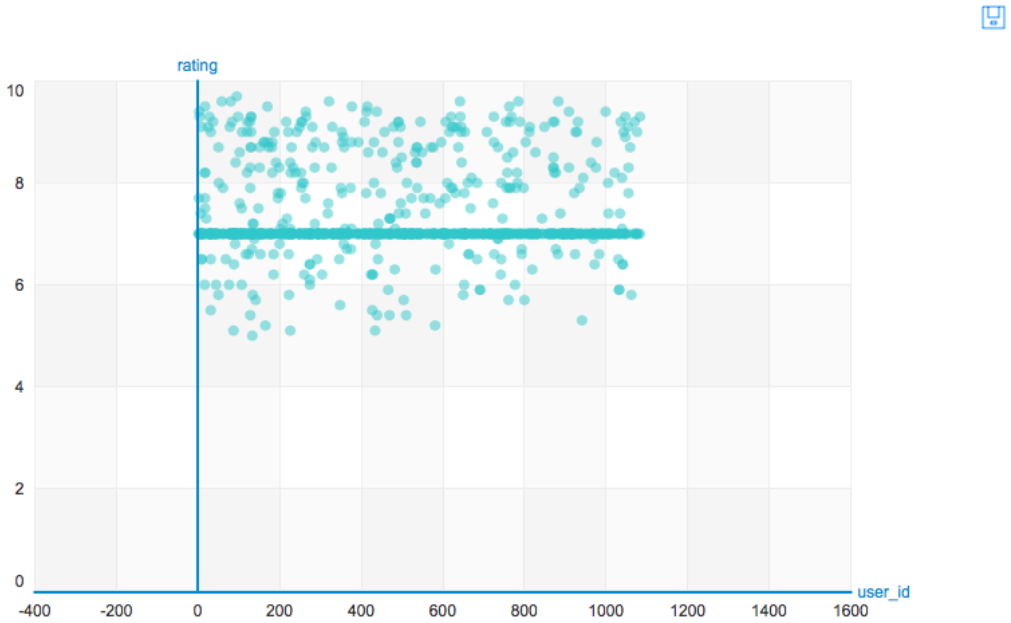

Figure 4.2 – Ratings of users on a scale of 0 - 10

Figure 4.1 represents the checkin counts in three different timeframes of the day, '1' for 'Morning', '2' for 'Afternoon', and '3' for 'evening'. Figure 4.2 plots the rating scale of 0 - 10 and the ratings of the users. The median value of the ratings is 7 according to the graph. Foursquare dataset does not provide ratings directly from the work of [YZZY15]. We used the Foursquare Developer API to obtain the ratings using the "venue_id" from the dataset. We were

successful in obtaining only the aggregated ratings for a subset of venues in our data and not all the venues, as ratings are not available for all the venues on Foursquare.

Using this dataset of Foursquare, we modeled the Tensor *pazocalR* based on ExIm model by using Fuzzy rules and matrix to fill the tensor with values from Fuzzy matrix. In real time evaluation, instead of filling values of fuzzy matrix into the tensor. We can directly multiply the fuzzy matrix values with the frequency of user visits to the place and store it in the tensor. Every new visit to the place would update the tensor values for that user if the contextual situation matches a previous contextual situation. But in offline evaluation it is not possible to do so, hence fuzzy matrix values can be considered directly.

Listing the problems of datasets

- Aggregated Ratings
- Aggregated checkins
- Privacy policies

Solutions

- Diverse dataset with more values.
- Diversity in temporal data.
- More venues with checkin data.
- Active users data.

Results

Best performing algorithms with Higher *Precision* and *Recall* values would be the best choice of algorithm for POI recommender system with heterogeneous data.

In our work reported in this document, the evaluations would not be accurate when we consider aggregated ratings to be personal ratings of the users. Inspite of this, we planned to conduct evaluations ignoring the aggregated ratings and considering them as personal ratings of the users.

Although we tried to gather Aggregated ratings from Foursquare using their API and process the existing Foursquare dataset. Only **22,322** timestamps had the ratings values and the rest were empty rating values. Because the dataset before filtering and ratings as mentioned above had **1048** users, **2445** venues and **143246** timestamps, and after filerting had only **22,322** checkin values with both timestamps and ratings out of 143,246 total values.

Major problem was:

number_of_ratings *equals* number_of_timestamps i.e. 22,322 values.

Hence, the time dimension of the tensor became useless and tensor factorization failed to work and provided wrong results. We compared the working of ExIm model on which did not perform well as the dataset had unique values for timestamps and ratings for each user, we used

models from [XCH+10, KABO10, PTC14, KCL15]. Because the results were not good, we decided not to tabulate the results, and perform evaluations on another dataset which we would create in the upcoming days. Hence we propose an evaluation procedure for a dataset which we plan to generate in the upcoming days using Librec tool [GZSYS15]. We detail the procedure of *user study* and *offline evaluation* and list the steps we plan to perform to generate the dataset.

4.3 Proposed Evaluation and Dataset Generation

Evaluating a proactive recommender needs real-time data and suitable feedback from the same set of users who visit the places and receive recommendations. Hence it is important to have the same set of users providing both the rating and checkin data. Dataset which we plan to generate would be done in the following steps:

- Users would be asked to use a mobile application.
- Application would store the GPS tracks of the user after the user leaves home, until he returns back.
- Users would be then asked to select the right place visited by them based on a list of recommended places.
- Selected place by the user would be asked to rate about her experience.

We would then use the procedure used by [FCRS13] to extract the stay points and use Foursquare API to find the nearest places. These places would be provided to the users to choose the visited place and rate it once she returns to her home. By this approach we gather, implicit data with the help of GPS tracks, timestamps, ratings, activities done by the user on that day. This dataset can be split easily into training and testing data and tested for precision and recall values.

This method facilitates to know the real annotations for the dataset collected by the mobile application, as done by [BCP14] in annotating the data generated by the users. As users annotate the data they generated, we also can store the ranks of places chosen by the user to represent her visit at that place and actively learn her preference after the initialization of the application as discussed in [ERR14]. Hence this facilitates to generate the dataset, as well as perform evaluation.

— 5 —
Conclusion

The availability of data and its effective use can lead to many useful applications. We discuss one possibility, which is proactive recommender systems. We detailed several types of explicit and implicit feedback concerned to travel domain, fuzzy logic for combining both the feedback, matrix and tensor factorization approaches for recommendations.

We discuss the intuition behind using location and time as context for recommendations. Proactive recommender systems are one of the upcoming class of systems which work on the basis of contextual factors. At the right contextual situation, the right set of places are recommended to the right user. This can be applied in mobile scenarios to deliver suitable recommendations while the user is on the move. To support such system, the context has to be analysed and modeled suitably. Building a new algorithm to model data for a new system or a new domain is not productive, instead one can focus on modeling the data in suitable manner for the existing state-of-the-art algorithms.

Several algorithms are used for providing context aware recommendations but they often use homogeneous data [YSQ$^+$15]. We discuss the advantages and also difficulties in using heterogeneous data for context-aware recommendations. We propose a fuzzy logic model to combine explicit and implicit input data (heterogeneous data) into a fuzzy output data which is fed into a tensor. The tensor is decomposed from a sparse tensor to smaller rank matrices. The dot product of these matrices can provide the estimated values of data. For a new user entering the system, similarity measures such as Jaccard similarity can be used to compute the right contextual situations (instead of rating values) and provide the recommendations suitable at that context.

The tensor factorization model (ExIm) can be modified to take into account different types of input for different domains of recommender systems. Hence this can support proactiveness for recommendations.

5.1 Future Work

We plan to build an application to collect data and annotate the places visited with the help of some users which will facilitate better evaluations. ExIm Tensor model can be improved and modified for the large scale recommender systems for various domains which employ deep learning algorithms for learning various anomalies in the data. These anomalous situations can depict uncommon behavior which might be useful to detect uncommon contextual situations in a large scale dataset.

Bibliography

[AM06] Sarabjot Singh Anand and Bamshad Mobasher. *Contextual recommendation*. Springer, 2006.

[AT11] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2011.

[BCP14] David Blachon, Doruk Cokun, and François Portet. On-line context aware physical activity recognition from the accelerometer and audio sensors of smartphones. In *Ambient Intelligence*, pages 205–220, 2014.

[BCR14] Matthias Braunhofer, Victor Codina, and Francesco Ricci. Switching hybrid for cold-starting context-aware recommender systems. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 349–352. ACM, 2014.

[BK07a] Robert M Bell and Yehuda Koren. Lessons from the netflix prize challenge. *ACM SIGKDD Explorations Newsletter*, 9(2):75–79, 2007.

[BK07b] Robert M Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 43–52. IEEE, 2007.

[Bla06] P Black. Manhattan distance. dictionary of algorithms and data structures. *US National Institute of Standards and Technology.*, 31, 2006.

[BLR11] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. Matrix factorization techniques for context aware recommendation. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 301–304. ACM, 2011.

[Bur07] Robin Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer Berlin Heidelberg, 2007.

[DBSR15] Nofar Dali Betzalel, Bracha Shapira, and Lior Rokach. ”please, not now!”: A model for timing recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys ’15, 2015.

[Dey01] Anind K Dey. Understanding and using context. *Personal and ubiquitous computing*, 5(1):4–7, 2001.

[ERR14] Mehdi Elahi, Francesco Ricci, and Neil Rubens. Active learning in collaborative filtering recommender systems. In *E-Commerce and Web Technologies*, pages 113–124. Springer International Publishing, 2014.

[FCRS13] Barbara Furletti, Paolo Cintia, Chiara Renso, and Laura Spinsanti. Inferring human activities from gps tracks. In *Proceedings of the 2Nd ACM SIGKDD International Workshop on Urban Computing*, UrbComp '13, 2013.

[FKCC+15] Marie-Christine Fauvet, Sanjay Kamath, Isaac-Bernardo Caicedo-Castro, Pathathai Na-Lumpoon, Ahmed Lbath, and Lorraine Goeuriot. Offering context-aware personalised services for mobile users. In *International Conference on Service-Oriented Computing*, pages 495–498. Springer Berlin Heidelberg, 2015.

[GZSYS15] Guibing Guo, Jie Zhang, Zhu Sun, and Neil Yorke-Smith. Librec: A java library for recommender systems. *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 23rd International Conference on User Modeling, Adaptation and Personalization*, 2015.

[KABO10] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

[KB09] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[KCL15] Volodymyr Kuleshov, Arun Tejasvi Chaganty, and Percy Liang. Tensor factorization via matrix factorization. *arXiv preprint arXiv:1501.07320*, 2015.

[KGF16] Sanjay Kamath, Lorraine Goeuriot, and Marie-Christine Fauvet. Processing natural language queries to disambiguate named entities and extract usersǵoals: application to e-tourism. 2016.

[Kor08] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[LC16] Gai Li and Qiang Chen. Exploiting explicit and implicit feedback for personalized ranking. *Mathematical Problems in Engineering*, 2016, 2016.

[LS01] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[LXZY10] Nathan N. Liu, Evan W. Xiang, Min Zhao, and Qiang Yang. Unifying explicit and implicit feedback for collaborative filtering. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, 2010.

[Men95] Jerry M Mendel. Fuzzy logic systems for engineering: a tutorial. *Proceedings of the IEEE*, 83(3):345–377, 1995.

[MR09] Tariq Mahmood and Francesco Ricci. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82. ACM, 2009.

[MZ09] Benjamin M Marlin and Richard S Zemel. Collaborative prediction and ranking with non-random missing data. In *Proceedings of the third ACM conference on Recommender systems*, pages 5–12. ACM, 2009.

[NDB07] An-Te Nguyen, Nathalie Denos, and Catherine Berrut. Improving new user recommendations with rule-based induction on cold user data. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 121–128. ACM, 2007.

[PTC14] Anh-Huy Phan, Petr Tichavsky, and Andrzej Cichocki. Deflation method for candecomp/parafac tensor decomposition. In *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, pages 6736–6740. IEEE, 2014.

[RRS11] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer US, 2011.

[RV97] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[SG11] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[SKB+13] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, and Alan Hanjalic. xclimf: optimizing expected reciprocal rank for data with multiple levels of relevance. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 431–434. ACM, 2013.

[SM08] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008.

[SM11] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. NIPS Proceedings, 2011.

[Tan97] Kazuo Tanaka. An introduction to fuzzy logic for practical applications. 1997.

[WHBGV11] Wolfgang Woerndl, Johannes Huebner, Roland Bader, and Daniel Gallego-Vico. A model for proactivity in mobile, context-aware recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 273–276. ACM, 2011.

[WWM75] Noah McKechnie Webster, Noah Noah Webster, and Jean L Mckechnie. *Webster's new twentieth century dictionary of yhe english language unabrid ged.* Number C/423 W4. 1975.

[XCH+10] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G Schneider, and Jaime G Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the 2010 SIAM International Conference on Data Mining*, Colombus, Ohio, US, 2010.

[YSQ+15] Lina Yao, Quan Z. Sheng, Yongrui Qin, Xianzhi Wang, Ali Shemshadi, and Qi He. Context-aware point-of-interest recommendation using tensor factorization with social regularization. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, 2015.

[YZZY15] Dingqi Yang, Daqing Zhang, Vincent W Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, 45(1):129–142, 2015.

[ZX10] Yu Zheng and Xing Xie. Learning location correlation from gps trajectories. In *MDM 2010*. IEEE, May 2010.

Fuzzy logic

Complete set of fuzzy rules for generation of combined feedback values

- If (Explicit is Very_Good) and (Implicit is Crowded) then (Tensor_Data is 1)
- If (Explicit is Very_Good) and (Implicit is Moderate) then (Tensor_Data is 1)
- If (Explicit is Very_Good) and (Implicit is Empty) then (Tensor_Data is 1)
- If (Explicit is Good) and (Implicit is Crowded) then (Tensor_Data is 1)
- If (Explicit is Good) and (Implicit is Moderate) then (Tensor_Data is 1)
- If (Explicit is Good) and (Implicit is Empty) then (Tensor_Data is 1)
- If (Explicit is Average) and (Implicit is Crowded) then (Tensor_Data is 2)
- If (Explicit is Average) and (Implicit is Moderate) then (Tensor_Data is 2)
- If (Explicit is Average) and (Implicit is Empty) then (Tensor_Data is 2)
- If (Explicit is Bad) and (Implicit is Crowded) then (Tensor_Data is 3)
- If (Explicit is Bad) and (Implicit is Moderate) then (Tensor_Data is 3)
- If (Explicit is Bad) and (Implicit is Empty) then (Tensor_Data is 3)
- If (Explicit is Worst) and (Implicit is Crowded) then (Tensor_Data is 3)
- If (Explicit is Worst) and (Implicit is Moderate) then (Tensor_Data is 3)
- If (Explicit is Worst) and (Implicit is Empty) then (Tensor_Data is 3)