Decision Support and Business Intelligence

Master Thesis

Molood ARMAN

Context-Aware Implicit Feedback based Hotel Recommender System for Anonymous Business Travellers

---

prepared at Amadeus/Hotel and BI distribution department

March, 2018 August, 2018

Confidential Thesis

Advisors:		
Srudeep KATAMREDDY	Amadeus, Nice	srudeep.katamreddy@amadeus.com
Christophe BLAYA	Amadeus, Nice	christophe.blaya@amadeus.com
Supervisor:		
Nacera BENNACER SEGHOUANI	CentraleSupelec	nacera.bennacer@supelec.fr

Acknowledgments

My deepest gratitude to the department responsible, Bernard Rannou, for considering me as a candidate to have this unique opportunity to be part of an awesome group, Special appreciations to my advisors Srudeep Katamreddy and Christophe Blaya for helping me to develop my professional skills and guiding every decision I have made. Thanks to all Amadeus Hotel BI and distribution team.
Thanks to the IT4BI Master programme: Esteban Zimanyi, Ralf Kustche, Toon Calders, Hatem Haddad, Patrick Marcel, Francesca Bugiotti and all the professors that inspired me with their knowledge and enthusiasm, specially to Nacéra Bennacer for her experiences and scientific advises on this master thesis. To the three universities that were part of the master: École Centrale Paris, Université Libre de Bruxelles and François Rabelais de Tours, excellent academically and rich culturally, giving me the possibility to meet wonderful people from around the world.
This journey would not have been possible without the support of my family and friends. To my family, thank you for encouraging me in all of my pursuits and inspiring me to follow my dreams. To my sisters and my friends, thank you for listening, offering me advice, and supporting me through this entire process. Special thanks to Afrooz, Azadeh and Forough who gave me energy everyday to continue this path.

Contents

Chapter 1

Introduction

One of the most widely discussed concepts in marketing analytics is "Cross-selling". Cross-selling is an old and valuable technique used by salespeople to increase order size and to transform single-product buyers into multi-product ones [Kamakura 2008]. It is a technique that involves the sale of multiple products offered by a single product/service provider to a new or existing customer. One of the most widely used cross-selling is when customers successfully book a flight, then they have the possibility to book a hotel in that destination among the hotels offered by the same flight booking system. Since the interest of customers are different from one to anther, the recommended hotels should be personalized based on their interest and since we cannot show all the results, the top hotels which chose to be shown to the customer should be more close to his interests rather than other related hotels. In this project, I just tried to implement a recommendation system in this way.

1.1 Motivation

Despite the increasing investments in cross-selling domain, firms find that these million-dollar marketing campaigns are not profitable [Li 2011]. A challenge was made to improve the response rates of a cross-selling campaign while avoiding the targeting of customers with irrelevant products/services. The use of recommendation systems became an important part of this challenge. They represent a powerful method for enabling users to filter through large information and product spaces. Nearly two decades of research on recommendation systems have led to a varied set of algorithms and a rich collection of tools for evaluating their performance. The field is moving in the direction of a richer understanding of how recommender technology may be embedded in specific domains. The differing personalities exhibited by different recommender algorithms show that recommendation is not a one-size-fits-all problem [Ekstrand 2011]. Specific tasks, information needs, and item domains represent unique problems for recommender systems, and design and evaluation of recommender systems needs to be done based on the problem requirements. Defining the problem correctly and use the correct algorithm for designing the recommendation system is still one of the most important tasks in the related field to increase the cross-selling profit.

On the other hand, business travelers have a huge economic impact and a very considerable contribution to the development of world trade and commerce and is a market that in most countries accounts for between one-quarter and one-third of all travel-related spending [Davidson 2003]. Amadeus as a provider of GDS (Global Distribution System) for numerous travel agency networks has a hotel recommendation service, which recommends a list of few hotels when a client reserves a flight. This service takes the travel

context (such as journey dates and location), sentiment data about hotels, hotel configuration data and hotel promotion subscriptions and by using a weighted sum model (WSM) recommends hotels to its customers. However, this service is more oriented toward leisure travel and does not target business travel segment. Business travel segment, managed by Travel Management Companies (TMC), comes with additional constraints like taking travel policy of companies into account which includes negotiated prices, hotel categories, maximum room rate, brand preferences and etc. We need a different approach rather than the current method to handle this case.

1.2 Context

1.2.1 Amadeus

Amadeus is a company dedicated to the world's travel industry. It's present in 195 countries. Amadeus was founded in 1987 by four European airlines: Iberia, Air France, Lufthansa and SAS as an independent and neutral global distribution system, to enable airlines to sell seats easily, more cost effectively and with a greater reach. This was the beginning of a journey to shape the future of travel. Amadeus offers cutting-edge technology solutions that help key players in the travel industry succeed in their business. Customers include travel agencies - both offline and online, airlines, airports, corporations, hotels, railway companies, ferry and cruise lines, car rental companies, ground handlers, and insurance companies. Amadeus success was based on being an efficient sale's system for travel providers and a source of comprehensive choice for travel sellers. Amadeus is the leading Global Distribution System (GDS) competing with Travelport and Sabre. This company has built a platform connecting together the travel industry agents in order to facilitate the distribution of travel products and services through IT solutions.

1.2.2 Business Vocabulary

1. **Business travel** is travel undertaken for the purpose of transacting and advancing the companies' business, and must only be undertaken where there is a demonstrated business need.

2. **Travel Management Companies (TMC)** is a travel agent consortium that fully manages the business travel requirements delegated by an individual, company or organization with the fundamentally save customers both time and money. A TMC is also known as a business travel agency or corporate travel provider. As opposed to the traditional travel agent, which will usually deal with leisure travel needs, a travel management company provides ongoing services with the aim of providing cost savings, keeping control of a travel policy and allowing the client to spend less time on time-consuming travel arrangements. A travel management company will typically use a GDS such as Saber or Amadeus to display real time availability to book flights, hotels, cars and trains.

3. **Business Policy** or travel policy is an arrangement usually made between travel agencies and companies in such a way as to:

   - minimize the overall cost of the trip;
   - increase the overall number of trips the traveler is going to make through the specific travel agency

   Passenger Name Record (PNR) is a record which stores the booking information. The reservation stored can belong to multiple passengers, as long as they all have the same itinerary. The information contained by a PNR includes the following items but is not limited to them: Passenger information (Name, email addresses, Contact information and etc), Segments (origin, destination, flight ID, departure and arrival date), Services (additional luggages, seat, special meal, booking cars and hotels), Travel agency information (Office Id) and etc. one example of PNR is shown in figure 1.1

```
> er

--- RLR ---
RP/NCE1A016U/NCE1A016U            FJ/SU  14FEB17/1735Z   3TOC3B
  1.FANI/JEAN MR
  2  AF6203 Y 20JUN 2 NCEORY HK1  0630 0755  20JUN  E  AF/3TOC3B
  3 AP 999999999999
  4 TK TL15FEB/NCE1A016U
  5 OPW-24FEB:1100/1C7/AF REQUIRES TICKET ON OR BEFORE
        27FEB:1100/S2
  6 OPC-27FEB:1100/1C8/AF CANCELLATION DUE TO NO TICKET/S2
  7 RM TEST AMADEUS
  8 RM TEST
```

Figure 1.1: An example of PNR

On the other hand, there is another contract between travel agencies and hotels which define certain requirements between them and usually defines information such as commission percentage on the total price of the rooms in hotels booked through the specific travel agencies.

1.3 Problem Statement

In order to enhance Amadeus relationship with its customers, an algorithm developed by hotel and BI distribution team called Hotel Optimizer, a multi-criteria search algorithm based on weighted sum model that recommends five hotels to any kind of travelers (business or leisure). These recommendations are provided through travel agencies which use GDS to book a flight for travelers. The purpose of this algorithm is to mix and weigh many different parameters such as distance, price, sentiment, star rating, sponsorship and commission level for the travel agencies and propose some hotels which fit better these specific features for each specific traveler. This algorithm considers both leisure and business travelers at the same time and try to make a difference between these two types by giving different weights to available criteria.

Attribute	Booking Class	Traveler type	Weight
Stars	Considering	both	different
Distance from airport	Not Considering	both	same
Distance from Center	Not Considering	both	same
Service	Considering	leisure	-
Room Comfort	Considering	both	different
Internet	Not Considering	both	different
Nationality	Not Considering	leisure	-
Sentiment	Not Considering	both	different
length of stay	Considering	both	different
Price	Considering	both	different

Table 1.1: A summary of features for Hotel Optimizer

Although there is a limitation for business travelers, this algorithm is satisfying leisure travelers since most considered criteria are associated with leisure context such as distance from center, having swimming pool, sentimental opinion about price and services by considering long stay for families and people with children. For Business travelers, there is a travel policy among companies, travel agencies and Hotels. These travel policies are unknown to Amadeus and they did not consider it in Hotel Optimizer. There are not any standards for these travel policies because each TMC defines their own policies to work with different companies and with different hotel chains in different cities. The main step for providing good recommendations for business travelers should be understanding these unknown business policies. The main problem for this project is finding the travel policy. A summary of these current features in Hotel Optimizer, where their weights are changing by different booking class of flights (very wealthy, wealthy, cost conscious and very cost conscious), their traveler types and whether they have same weights in different traveler types or not can be seen in table 1.1.

Based on table 1.1, it's obvious that a feature like distance from airport in current hotel optimizer system has the same weight for different travelers type (both leisure and business travelers). The weight of this feature is not changing based on different booking class of the flights (Not Considering).
Except the sentiment information which is a global rating based on average opinion of the users, room comfort, service and Internet which are extracted from reviews, all other information are extracted from PNR. The information extracted from reviews are known as E-reputation features which includes room comfort, sleep quality, location, facility, service, staff, Internet and swimming pool. Each feature of E-reputation can receive a score between 0 to 100 per each hotel.
Some penalties was considered for price and far distances and also for bad chains and low star hotels to decrease the possibility of having bad hotels in the final list of recommendations. The weights is defined by experts and was modified after several tests and by changing the weights, the results can be easily modified. The final result of this system is the hotel with highest score and a set of five alternative hotels with next higher scores based on this method. For example, for a traveler with booking class Y (cost conscious) to city Paris, the result of this system is shown in figure 1.2.

As a result, the next phase for the Amadeus company is to improve the recommendation and try to work on how to take into account the preferences of travel agencies for

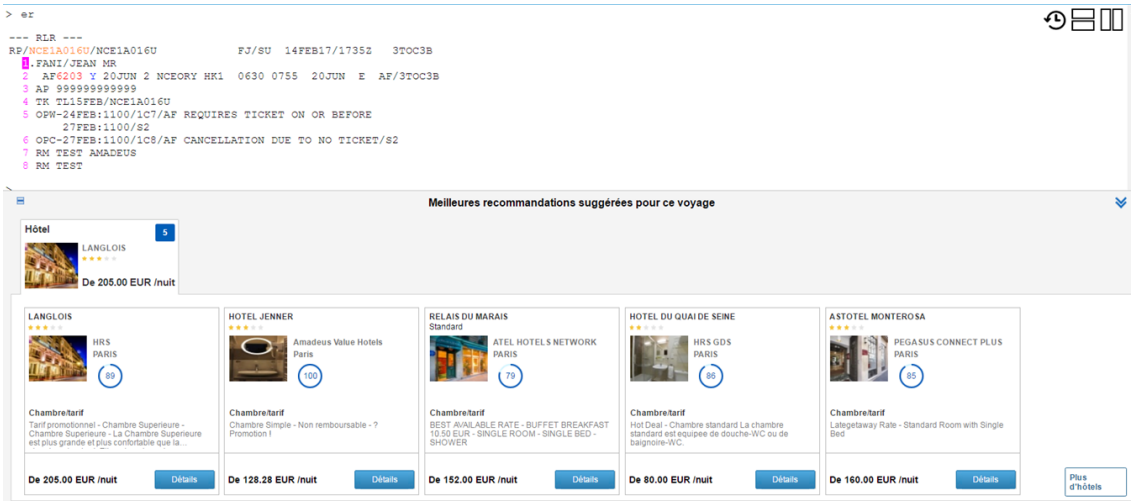


Figure 1.2: Recommended hotels based on the current recommendation system _ hotel Optimizer

business travelers.

There are three problems for recommending hotels to business travelers by the current system_ Hotel Optimizer_ so far:

1. Any kind of information related to business policy between companies and hotels are not considered in this method. For example, the price boundaries for sure is part of business policy as each company defines a price range in which it expects its employees to book a hotel for their business travels. This price is just known by the employees of a specific companies and is written in the defined business policy of each company.

2. The e-reputation features reflect the opinion of both leisure and business travelers, but it's not clear which these business travelers are TMCs customers or not.

3. In the final list of current recommended hotels for a business traveler, the prices and other features of the hotels that are out of the travel policy associated to a company can be seen.

Also, as we know, traditional recommender systems operates in the two-dimensional $User \times Item$ space; because of that, they make their recommendations based only on the user and item information and do not take into the consideration additional contextual information that may be crucial in some applications. However, in many situations the utility of a certain product/service to a user may depend significantly on other parameters like time (e.g., the time of the year, such as season or month, or the day of the week). It may also depend on the person with whom the product will be consumed or shared and under which circumstances. In such situations it may not be sufficient to simply recommend items to users; the recommender system must take additional contextual information, such as time, place into consideration when recommending a product [Adomavicius 2005b]. For example, when recommending hotels to a traveler, the

system should also consider the place of those hotels to be in the same city as destination city of the traveler, budget and time can be other contexts which can be considered as traveling conditions and restrictions and many other contextual information. Here, for each traveler, the relation of the company, travel agency and the hotels which has the contract with that travel agency should be taken into account. It is important to extend traditional two dimensional $User \times Item$ recommendation methods to multidimensional settings. In addition, in several researches, it was shown that considering knowledge about users into the recommendation algorithm in certain applications can lead to better recommendations [Herlocker 2000].
To that end, this thesis focuses to design a hotel recommendation system for the anonymous business travelers when they reserve a flight.
They are anonymous because even in the booking data, the company which booked these hotels are not mentioned. Also, gathering information about companies without their permission is not allowed. Then, the first step should be applying a method to find the related company to each transaction. Since our target group is business travelers, we should consider a group of customers from the same company as one user. Then actually for finding a user, we have to cluster transactions from the same company in one group.
Also, the only feedback which we have about interest of companies to book hotels are derived from historical data of booking and we do not have access to the direct ratings of users about hotels. Then we should use the methods which have a better performance for implicit feedback data.

1.4 Objectives and Contributions

The main goal of the project is to provide a feasibility study of a potential solution based on the historical data related to hotel booking to provide the best possible recommendation systems for anonymous business travelers. That means developing a methodology for hotel BI and distribution department to recommend hotels to the business travelers through TMCs. The used data is just historical booking data since sentiment analysis information. Then we just decide to use historical data of booking to the aim of this project. The utility matrix as the base of the recommendation system is built by considering the number of bookings for a hotel by a company in the past. The number of companies and hotels in our data set are huge, then the final utility matrix is very sparse. As you know using collaborative filtering method are not working fine in case of high sparsity.

Moreover, the second major objective is to define the business policy. We tried some methods which I explain in section 3.2 based on the constraints and limitations of the data but still the result of this section partially defines the unknown business policy. It's the part which are clearly observable for us, but still business policy can be defined based on lots of other unknown features, then one method for exploring them is extracting them from the same patterns of historical bookings. One way can be just finding the latent factors for companies and defines them as business policy from company side. The same holds for hotel side. We should just find a method to combine the extracted features from known business policy with latent factors as the unknown features of the business policy.

The final approach which we implement to reach these goals for this project guides us to these deliverables:

- Developing a model-based recommendation system which can work just based on implicit data;
- Creating the profiles per each user and item by just using the implicit data and the interaction of the users with items based on the transactions data;
- Having a better performance by embedding these created profiles in the learning process of model-based recommender system;
- Considering context to give more accurate recommendations to each customer.

The main steps to reach these objectives can be seen in 1.3

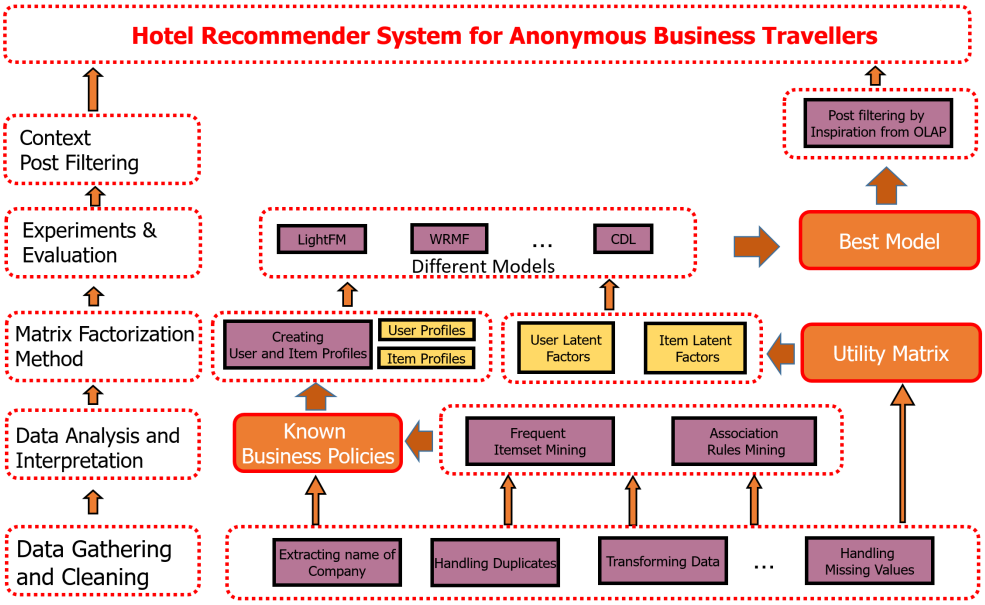


Figure 1.3: Methodology to build a hotel recommender system

1.5 Thesis Outline

The remainder of this thesis is organized as follows: Chapter 2 discusses related work about our problem. The data processing, cleaning, analysis and exploration is presented in Chapter 3. Chapter 4 and 5 cover the methods which we used to solve the problem. In chapter 4 we propose our first solution, In chapter 5, our final solution is proposed. Chapter 6 provides the different experiments based on the proposed methods of chapter 5. Chapter 7 provides the evaluation method to choose the best model for our problem, and finally, chapter 8 concludes the work presented in this thesis and mentioning future works.

Chapter 2

Recommendation Systems

Personalization of product/service information has become one of the most important factors that affect the selection and satisfaction of the customers in today's competitive and challenging market. Personalized service requires firms to understand customers and offer goods or services that meet their needs. Successful firms are those which provide the right products to the right customers at the right time and for the right price. Recommendation systems have become an important research topic in information sciences and decision support systems and are widely used by e-commerce practitioners [Yao 2015]. Recommendation systems are decision aids that analyze customer's prior online behavior and present information on products or services to match customer's preferences. Hotels are one of the businesses which try to increase the number of bookings of their rooms by customers. Some researches are already done in this field to implement hotel recommendation systems by using different machine learning algorithms [Nilashi 2017, Shenoy 2017, Mavalankar 2016].
In this chapter, firstly, I review the different models of recommendation systems and the current challenges of these models. Also, I discuss about data acquisition to design the recommendation systems which results to apply different methods. Then, we review some research that just focuses on hotel recommendation systems. Also, I explain a previous master thesis which was done by an intern at Amadeus for the same problem, in 2017, with the aim of collecting, exploring and analyzing booking data in order to build a decision support tool with machine learning algorithms. I explain what was his approach and what was the drawbacks of his work. Finally, different methods of evaluation for recommender systems are discussed.

This section first presents different types of traditional recommendation systems (RS). Then we describe multi-dimensional RS as the new generation which context-aware recommendation systems are part of them. There are different classification of RS but the most widely used classification divides the filtering algorithms into [Adomavicius 2005b, Candillier 2007, Ekstrand 2011]: (a) collaborative filtering, (b) demographic filtering, (c) content-based filtering and (d) hybrid filtering.

2.1 Content-based filtering

This model of RS makes recommendations based on user choices made in the past. Content-based (CB) filtering also generates recommendations using the content from objects intended for recommendation; therefore, certain content can be analyzed. From this analysis, a similarity can be established between objects as the basis for recommending items similar to items that a user has bought, visited, heard, viewed and ranked positively

[Bobadilla 2013]. The CB recommender systems executes these two steps as their basic principle to recommend an item to a user : 1) analyzing the description of the items preferred by a particular user to determine the principal common attributes (preferences) that can be used to distinguish these items. These preferences are stored in a user profile. 2) Comparing each item's attributes with the user profile so that only items that have a high degree of similarity with the user profile will be recommended [Lu 2015]. In CB recommender systems, two main techniques have been used to generate recommendations. One technique generates recommendations heuristically using traditional information retrieval methods, such as cosine similarity measure. The other technique generates recommendations using statistical learning and machine learning methods, largely building models that are capable of learning users' interests from the historical data (training data) of users.
The CB recommender systems have the ability to recommend new items even if there are no ratings provided by users. Also, if the user preferences change, it has the capacity to adjust its recommendations in a short duration of time. They can manage situations where different users do not share the same items, but only identical items according to their features. On other hand, it's necessary to know that these techniques suffers from some limitations: Content based filtering techniques are dependent on items' metadata. That is, they require rich description of items and very well organized user profile before a recommendation can be made to users. In the literature, this is called limited content analysis. So, the effectiveness of CB filtering methods depends on the availability of descriptive data. CB recommender systems have overspecialized recommendation problems. Users are restricted to have recommendations similar to items already defined in their profiles [Isinkaye 2015].

2.2 Demographic filtering

This model of RS is justified on the principle that individuals with certain common personal attributes (sex, age, country, etc.) will also have common preferences. We can consider this model as a kind of cluster-based recommendations just based on some defined groups.

2.3 Collaborative Filtering

Collaborative Filtering (CF) allows users to give ratings on a set of elements in such a way that when enough information is stored on the system, we can make recommendations to each user based on information provided by those users we consider to have the most in common with them [Bobadilla 2013]. There are two main approaches to collaborative filtering: (a) memory-based (usually known as neighborhood approach) and (b) model based approach. Memory-based methods are better at finding similarities in local relations, whereas model based approaches learn more general structures that cover a larger part of the data (mostly all of data) [Steeg 2015, Isinkaye 2015]. Collaborative Filtering has some major advantages over content based filtering as it can perform in domains where there is not much content related to items or when content is not easy to analyze through

technical methods, but some methods like sentiment analysis, opinion mining and etc. is required.

2.3.1 Memory-based Collaborative Filtering

Collaborative filtering algorithms that compute similarities among neighbors in memory are called memory-based CF. Memory-based methods can be defined as methods that require all ratings, items, and users be stored in memory [Schafer 2007]. Memory-based methods usually use similarity metrics to obtain the distance between two users, or two items, based on each of their ratios [Bobadilla 2013]. This method predicts the ratings based on this kind of similarity. The majority of these methods can be divided in two different parts: 1) User-user collaborative filtering; 2) Item-item collaborative filtering.

It's necessary to keep in mind that pure memory-based models do not scale well for real world applications. Thus, almost all practical algorithms which are useful to solve the real-world applications use some form of pre-computation to reduce run-time complexity. As a result, the current practical algorithms are either pure model based algorithms or a hybrid of some pre-computation combined with some ratings data in memory [Schafer 2007].

2.3.1.1 User-User Collaborative Filtering

User-user collaborative filtering is one of the first known CF methods. It was first introduced in the GroupLens Usenet article recommender, the Ringo music recommender and the BellCore video recommender [Ekstrand 2011].The core introduction of the CF method in an algorithmic way is: find other users whose past rating is similar to the current user and use their ratings on different items to predict what the current user will like.
The rating matrix $R : User \times Items$, a user-user CF system requires a similarity function $S : U \times U \rightarrow R$ which computes the similarity between users and a method for using similarities and ratings to finally generate predictions [Ekstrand 2011].
This method is also known as KNN collaborative filtering because the most widely used algorithm for collaborative filtering is the K Nearest Neighbors (KNN) [Bobadilla 2013]. In the user to user case, KNN executes the following three tasks to generate recommendations for an active user: (1) determine k users' neighbors (neighborhood) for the active user A; (2) implement an aggregation approach with the ratings for the neighborhood in items not rated by A; and (3) extract the predictions in step 2 then select the top N recommendations.
However, User-user collaborative filtering has the scalability problems limitations as the user base grows [Ekstrand 2011]. To extend collaborative filtering to large user bases, it is necessary to develop more scalable algorithms.

2.3.1.2 Item-Item Collaborative Filtering

Item-item collaborative filtering also called item-based collaborative filtering is one of the most widely deployed collaborative filtering techniques. Item-item collaborative filtering has been used by Amazon.com in 2009 [Ekstrand 2011]. Rather than using similarities between users' ratings to predict preferences, item-item CF uses similarities between the rating patterns of items. If some items tend to have the same users' attention such as

like and dislike, number of bookings or etc., then they are similar and users are expected to have similar preferences for similar items. Therefore, this method is similar to earlier content-based approaches to recommendation, but the main difference is that item similarity is deduced from user preference patterns rather than extracted from item data [Ekstrand 2011].

2.3.2 Model-based Collaborative Filtering

Model-based algorithms are methods from which a model is derived and used to compute the recommendation. These methods periodically create a summary of ratings patterns offline [Schafer 2007]. This model tries to guess how much a user will like an item that it did not encounter before. For this aim, they utilize several machine learning algorithms to train on the vector of items for a specific user, then they can build a model that can predict the user's rating for a new item that has just been added to the system. Most widely used data mining and machine learning algorithms in this methods are Bayesian classifiers, neural networks, Regression and clustering, latent semantic methods and matrix factorization [Bobadilla 2013]. In comparing with model-based RS, memory based RS called 'lazy learning' methods since they do not build any model but instead they perform heuristic computation at the time of recommendations [Adomavicius 2005a]. Model-based methods analyze the utility matrix to identify relations between items; they use these relations to compare the list of top-N recommendations. Model based techniques solve the sparsity problems related to recommendation systems [Isinkaye 2015]. I explain latent factor models and matrix factorization as examples of this technique in the following:

2.3.2.1 Latent factor models

Latent factor models which are informally known as SVD models (Singular Value Decomposition) represent users and items as vectors in a common low-dimensional "latent factor" space [Isinkaye 2015]. Traditional latent ranking models score the i_{th} item $di \in R^D$ given a query $q \in R \times D$ using the following scoring function:

$$f(q, di) = q^T W di = q^T U^T V di \tag{2.1}$$

where $W = U^T V$ has a low rank parameterization, and hence $q^T U^T$ can be thought of as the latent representation of the query and Vdi is equivalently the latent representation for the item. The latent space is n-dimensional, where $n \ll D$, hence U and V are $n \times D$ dimensional matrices. This formulation covers different algorithms and applications [Weston 2012]. For example, in collaborative filtering, it is required to rank items according to their similarity to the user, and methods which learn latent representations of both users and items are very effective. In particular, Singular Value Decomposition (SVD) and Non-negative Matrix Factorization (NMF) are two standard methods that use equation 2.1 [Weston 2012].

2.3.2.2 Matrix Factorization

Some of the most successful realizations of latent factor models are based on matrix factorization. The essence of matrix completion technique is to predict the unknown values

of utility matrix through the user-item matrices. Basically, matrix factorization describes items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors prompts a recommendation. These methods have become popular in recent years by combining good scalability with predictive accuracy. Also, they offered much adaptability for modeling different real-world situations [Koren 2009]. I discuss about these models more in chapter 5.

2.4 Hybrid Filtering

Hybrid recommender systems have emerged by combining multiple algorithms [Ekstrand 2011]. This method combines different recommendation techniques to gain a better system optimization to avoid some limitations and problems of previously known recommendation systems. Combination of algorithms will provide more accurate and effective recommendations than a single algorithm as the disadvantages of one algorithm can be overcome by another algorithm. Hybrid recommender system can be divided into different classes as followings [Bobadilla 2013, Isinkaye 2015, Burke 2002]:

- **Weighted recommenders** combine the scores produced by several recommender techniques present in the system to generate a recommendation list for the user.
- **Switching recommenders** use some features to switch between different recommendation techniques and algorithms and finally use the algorithm expected to have the best result in a particular context.
- **Mixed recommenders** give the results of several different recommenders technique together. This seems similar to weighted recommenders, but the results are not necessarily combined into a single list.
- **Feature combining recommenders** use multiple different recommendation data sources as inputs. This method is well known for merging the content/collaborative recommenders to treat collaborative information as additional feature data associated with each example and use content-based techniques over this extended data set.
- **Cascading recommenders** use the output of one algorithm as the input of another recommendation system technique.
- **Feature augmenting recommenders** produce a rating or classification of an item with a recommender algorithm and that information is then used as one of the input features of the next recommendation technique.
- **Meta level recommenders** train a model using one algorithm and use that model as input to another algorithm. This differs from feature augmentation since in feature augmenting method, we use a learned model to generate features for input to a second technique; but in a meta-level hybrid method, the entire model is the input.

The different type and techniques of traditional recommendation systems can be seen in figure 2.1.

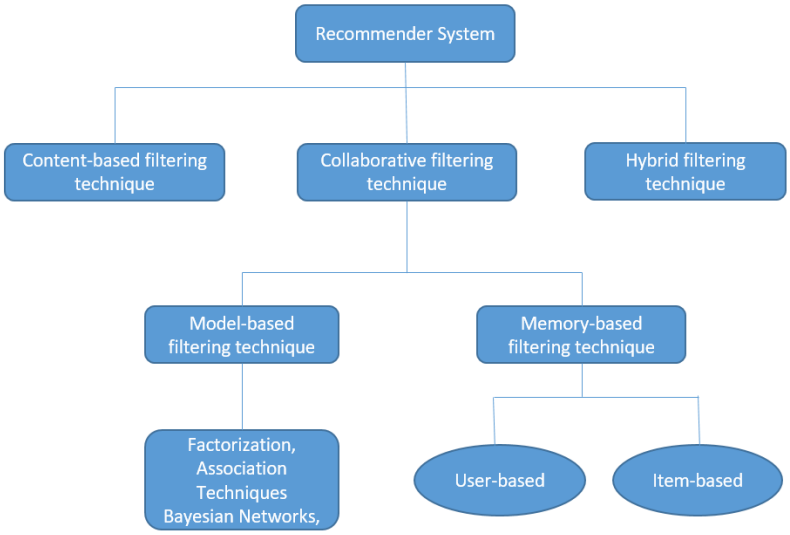


Figure 2.1: Recommendation Types and Techniques [Isinkaye 2015]

2.5 limitations and problems of Recommendation Systems

Various techniques used in a recommender system may experiences some of the challenges that are described in the following:

2.5.1 Sparsity

Sparsity problem is one of the significant problems of recommender system and data sparsity has great impact on the quality of recommendation result. This problem can occur when there is no access to enough information. The main reason behind data sparsity is that most users do not rate or use most of the items and the current ratings are usually sparse. Generally, data is represented in form of utility matrix of ratings given to items. When number of users and items increase, the matrix dimensions increase and sparsity evolves. Collaborative filtering suffers from this problem because it is dependent on the rating matrix in most cases. Many researchers have attempted to diminish this problem [Sharma 2013].

2.5.2 Cold Start

Cold start problem alludes to the situation when a new user or item just enters the system and there are not any current available data for them. New user problem, new item problem and new system problem are three different kinds of cold start problems. In such cases, it is extremely exceptionally hard to give recommendation as in case of new user, there is much less information about user that is available and furthermore for a new item, no ratings are usually accessible, then in this way, collaborative filtering cannot make helpful recommendations in case of new item and new user. However, content based

methods can provide recommendations in case of new item as they do not relies on any past ratings of other users to recommend the item [Sharma 2013, Bobadilla 2013].

2.5.3 Scalability

Scalability is the property of a system showing its capacity to deal with growing amount of data. With huge growth of information over Internet, it is obvious that the recommender systems are having an explosion of data and accordingly an extraordinary challenge made to handle with persistently growing demand. Some of the recommender system algorithms deal with increasing number of users and items which leads to grow the computations. In collaborative filtering, computations develop exponentially and get costly, and sometimes without having accurate results. Methods based on approximation mechanisms can be used to handle this problem and speed up recommendation computation. Regardless of improving performance, most of the time these methods result in accuracy reduction [Sharma 2013].

2.5.4 Over Specialization

Users are limited to getting recommendations which take after to those definitely known or defined in their profiles in some cases; This problem is called over specialization problem. It keeps users from discovering new items and other accessible choices. Diversity of recommendations is an alluring property of all recommendation system [Sharma 2013].

2.6 Implicit vs. Explicit Feedback

Recommender systems rely on different types of input such as the most convenient high quality explicit feedback, which includes explicit input by users regarding their interest in item or implicit feedback by inferring user preferences indirectly through observing user behavior.

2.6.1 Explicit Feedback

Data should be provided from user side. This data can include ratings, ranks, like or dislikes and etc. It requires efforts from the users and also, users are not always ready to provide enough information. Despite the fact that explicit feedback requires more effort from user, it is seen as providing more reliable data, since the user preferences is not extracted from actions, or inferring assumptions. It also provides transparency into the recommendation process to gain a slightly higher perceived recommendation quality and more confidence in the recommendations [Isinkaye 2015].

2.6.2 Implicit Feedback

User's preferences are automatically inferred by monitoring the different actions of users such as the history of purchases, using some services, navigation history, or spending time on some web pages, and etc. Implicit feedback reduces the burden on users and does not need any effort from user side by inferring their user's preferences from their behavior with the system. Despite this advantage, this data is less accurate. [Isinkaye 2015].

2.7 Multi-Dimensional Recommendation Systems

All the approaches described in this section so far were focused on the recommending items to users or users to items, but they did not consider additional contextual information such as time and place. A recommender system may recommend a different product to a user based on different time of the month or even day, morning or afternoon. In marketing, behavioral researches on the customer decision making can change the recommendation to the specific users based on his decision making strategies under different contexts. Multidimensional recommendation model (MD model) present recommendations based on the multiple dimensions and extends the traditional two-dimensional (2D) $Users \times Items$ model. Traditionally, recommender systems deal with application that have two types of the entities, Users and Items [Adomavicius 2005a]. Some different models were defined so far. In [Adomavicius 2001], a multidimensional model was proposed based on the multidimensional data model used for data warehousing and OLAP which has the capability of aggregation on different dimensions. Also, the field of context-aware recommendation systems is a new category which if we consider context as fully observable and static, it can be considered completely like a multi dimensional RS with OLAP model. More specifically, hierarchical structure of the contextual information can be mapped to an OLAP model.

2.8 Context-Aware Recommendation Systems (CARS)

In traditional recommender systems, the prediction process usually begins with the specification of the initial rating set. The ratings can be explicitly collected from the users or inferred by the RS implicitly. Then the RS computes the rating function R

$$R : User \times Item \rightarrow Rating \tag{2.2}$$

for all the (user, item) pairs that have not been rated by the users. In reality usually dataset is sparse. In the field of traditional RS, this is usually solved by applying collaborative methods with latent factors, such as matrix factorization. However, If we want to consider the context from the first step, these methods are not always effective. In CARS, the dataset becomes extremely sparse since the rating function R is considering three dimensions

$$R : User \times Item \times Context \rightarrow Rating \tag{2.3}$$

What is context?

In a very simple definition, Context is any information that can be used to characterize the situation of an entity. Context type can be divided in two categories as following:

- **Representative Context**: Fully Observable and Static
- **Interactive Context**: Non-fully observable and Dynamic

As you can see in fig 2.2, these two types of context were shown in the form of the knowledge of recommendation systems about the contextual factors and how this context

changes. We can classify the knowledge of a recommender system about the contextual factors into three categories: fully observable, partially observable, and unobservable. In other word, the change of context over time can be classified in two categories: Static, Dynamics.

How Contextual Factors Change	Knowledge of the RS about the Contextual Factors		
	Fully Observable	Partially Observable	Unobservable
Static	Everything Known about Context	Partial and Static Context Knowledge	Latent Knowledge of Context
Dynamic	Context Relevance Is Dynamic	Partial and Dynamic Context Knowledge	Nothing Is Known about Context

Figure 2.2: Different Context Type [Adomavicius 2015]

How to collect context?

It is exactly related to the knowledge of recommendation system about the contextual factors. If they are fully observable, they can be extracted in an explicit manner such as time or location of the shopping. If they are unknown for our recommendation system, they are latent factors and should be extracted in an implicit manner; for example recommendation system can build a latent predictive model, solve where unobservable context is modeled using latent variables.

Approaches of Context Incorporation

Approaches for CARS is divided into three groups [Adomavicius 2015, Rahman 2013]: contextual pre-filtering, contextual post-filtering and contextual modeling. The paradigms of these three groups are illustrated in Figure 2.3. The most recent researches lay stress on contextual modeling using regression models especially on factorization methods.

- **Contextual pre-filtering (PreF)**: which is also known as contextualization of recommendation input. In this approach information about the current context is used for selecting only the relevant set of data, and ratings are predicted using any traditional 2D recommender system on the selected data.
- **Contextual post-filtering (PoF)**: which is also known as contextualization of recommendation output. In this approach contextual information is not considered initially, and the ratings are exactly predicted by using any traditional 2D recommender system on the entire data. Then, the resulting set of recommendations is adjusted for each user using the contextual information. This adjustment can be filtering the ranked order of items based on a specific context or it can be removing the results which are not fit in a specific context.
- **Contextual modeling (CM)**: which is known as contextualization of recommendation function. In this approach contextual information is used directly inside the

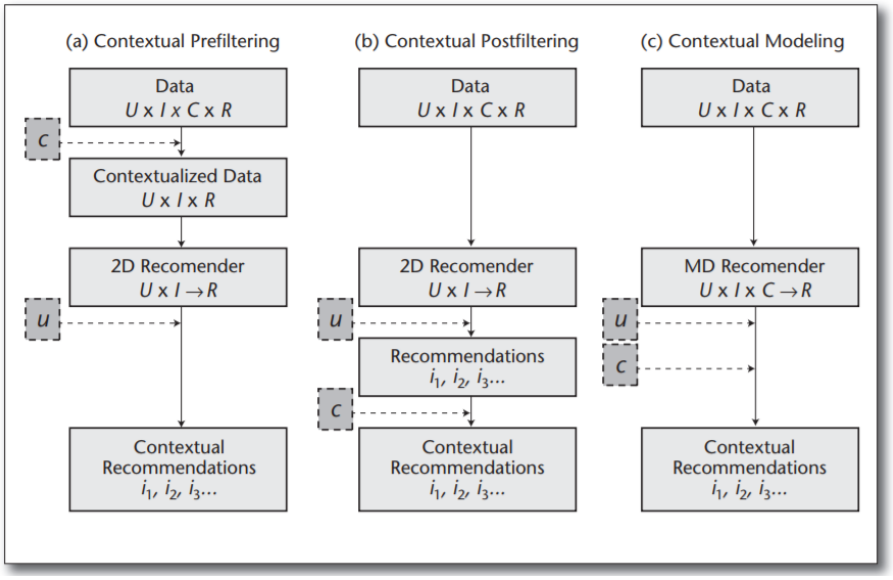


Figure 2.3: Paradigms for Incorporating Context in Recommender Systems [Adomavicius 2015]

model as part of the rating estimation. In this model, The traditional two dimension space of recommendation system extends to multi dimension spaces. Some specific models can be factorization learning or using hidden Markov models to contextualize the recommendation systems.

2.9 Hotel Recommendation Systems

Some research in the domain of the recommendation systems were done about recommending the hotels to the customers of a specific service. Expedia is a site with hundreds of millions of visitors every month; Once, it wanted to create a hotel search by providing personalized hotel recommendations to their users. The problem for this goal is that there aren't enough user data to personalize them for each user. They made a competition and in this competition, the task was contextualizing customer data and anticipating the likelihood which a user will stay at 100 different hotel groups. The train/test datasets used for this project have been given by Expedia, via Kaggle, and they contain 23 features capturing the logs of customer behavior. There is an additional dataset containing 149 features, which relates to the hotel reviews made by users. Kaggle's benchmark for arbitrarily speculating a user's hotel cluster is 0.02260, and the mean average precision K = 5 value for naive recommender systems is 0.05949. Some competitors like Gourav G. Shenoy and et al. [Shenoy 2017], they implemented the following algorithms:
1. Naive Bayes 2. Decision Trees 3. K Nearest Neighbors 4. K Means Clustering 5. Multinomial Logistic Regression 6. Ensemble Learning Methods with k-NN and Decision Trees.
The train dataset contains nearly 300 million records, while the test dataset contains nearly 75 million records. Expedia provided another additional 'destinations' dataset contains

nearly 62k records. In main dataset there are 100 clusters which are defined by Expedia. The data is well distributed over all 100 classes and there is skewness in the data. They predict a potential problem with so many classes in the data having 100 classes for the classification problem would result in a decrease of performance of the model. Hence, They have used a k-means clustering to group the data together. They created clusters with k=5, 10, 20 and 50. The dataset had multiple classes without any significant perceived pattern that relates them to the features. This initially made it hard to accomplish reasonable accuracy. After applying clustering and ensemble methods, they could accomplish noticeable increase in accuracy. The most astounding observations was by the Multinomial Logistic Regression, as it handles the numeric data efficiently. They proposed other algorithms like decision trees can work well in case of discrete data as well. Another group of competitor, Mavalankar and et al. [Mavalankar 2017] used multiple models and techniques to achieve the best solution. First, they used baseline as a method that uses heuristics, simple summary statistics or randomness to create predictions for a dataset. They used these predictions to measure the baseline's performance (for example, accuracy. MAP@5). This metric is what they compare other models performance against. Then, they used an ensemble of four different models (random forests, SGD classifier, XG Boost and Naive Bayes); XGBoost is preceded by completion of distance matrix which is an important feature in the dataset that is currently incomplete; then they applied data leakage solution which takes advantage of the fact that there is a potential leak in the provided data; and finally they applied a mixture of the methods of baseline and data leakage solution. To fill in the missing values in the dataset, they used matrix factorization technique. Also, they considered some useful information about which hotel cluster was finally chosen over other hotel clusters explored by the user. One important observation to note was that few users might be travel agents and could explore multiple types of destinations at the same time. This could also be true for a few users who are planning multiple vacations at the same time. That is why they considered the preferences of the users separately for each destination type he/she explored. Also, after a booking was made, subsequent searches by the user were treated separately. Furthermore, based on the details provided by Expedia on the competition page, hotels tend to change their cluster seasonally. To capture this tendency, one-hot encoding representation of the month was included from which user is seeking to start his/her stay. Finally, they made use of the latent features of the estimations provided in the dataset. However, since there are 149 latent features for each destination, they applied PCA to extract the most relevant dimensions. Hotel clusters do not seem to have strong linear correlation with any of the features. Thus, applying techniques that rely on modeling the linear relationship between features would be less fruitful. Hence, they decided to work mostly on tree based methods which are capable of capturing non-linear tendencies in data. Another reason for leaning towards tree based models was that most of the data provided was categorical which can be handled by tree based models without needing any conversion to one-hot encoding form. After applying the different models, it was concluded that Ensemble Learning with Data Leak model has the best result on the test data- a score of 0:496 on test data. This reaffirms this fact that combining several weak learners has a synergistic effect. Most of the methods which they used involved ranking of clusters by their predicted class probabilities which seems fair. Due to the volume of data, they could not replicate the distance matrix completion tech-

niques employed by the first rank solution. They considered it as the future improvement, wherein they can try replicating the existing code in conjunction with features, data leak solution and ensemble learning to achieve an even better result. Different ranking methods like RankSVM and RankBoost can perform well too but require a lot of resources.
Arruza M and et al. [Arruza 2016] used PCA to decrease the dimensions of the dataset. Then they used the naive Bayes conditional independence assumption to rank hotel clusters as the baseline method. This method outperformed the random benchmark put forward by Kaggle, and serves as their benchmark going forward for the relative success or failure of an algorithm. Another simple method used was training an SVM. One problem which they faced in applying SVMs was adapting the typically discriminative SVM algorithm to generate probabilities to be used to create a ranking of hotel clusters. They finally used a method described by Wu et al. relying on pairwise coupling of single class predictions, learning the class probabilities using cross-validation. One master thesis [AOULLAY 2017] was done last year with the same goal of current master thesis in the same department but the result is not satisfying. He did the ensemble methods (bagging, boosting and random forest) and he used the principal component analysis (PCA) as a dimensionality reduction technique and item-based collaborative filtering as the model for the recommendation system. He clustered the hotels based on their eReputation data (Room comfort, sleep quality, location, facilities, service, staff, Internet and swimming pool) which each feature had a score between 0 to 100. Despite the high precision he achieved for the machine learning method, which he applied, his recommendation system is not able to give a good result in reality. It is obvious that clustering based on the final product (hotels) was not practical here which defines the importance of understanding what we exactly need to cluster for this particular problem (travel agencies, companies (travelers), service packages or etc). In order to transform the recommendation problem into a classification problem, he considered the variable Y as the output equal to the name of the hotel. In this case Y may take different values. For large cities like Paris this number was quite large. Classification algorithm is not recommended in such situation. To solve this problem, first, the number of values that Y can take were decreased by grouping the hotels that are similar then a group of hotels was predicted instead of a hotel. Secondly, he tried to recommend hotels among the predicted group. He used unsupervised learning for clustering the similar hotels by using the eReputation data for clustering hotels and he tried the K-means method. He tried the algorithm with $k = 2$ or $k = 5$. The problem was the hotels in the same cluster had the same features but based on the eReputation data, two similar hotel could have the same sentiment data from the users but they would be completely different in the reality. Then he changed the output variable to the Y = Hotel Chain. In this new problem he dealt with imbalanced data. A reservation image is a snapshot of all the booking details and pricing information available at a certain given time. All the information about a booking and the states it goes through is being kept in a container of data called eVoucher. As the reservation evolves, the eVoucher database not only stores its evolutions but it also gives the possibility to retrieve all the images of each evolution. With each modification made to the booking, a new image of the reservation is produced and saved in the database. All the changes applied to a booking are preserved as the history section of the eVoucher, and they are available for each created image. He merged some eVoucher tables that were important to get all the booking information he

needed. Then he showed the final image of the hotel at the same time by the name of the recommended hotel in this way. As you get the idea, he did not consider the business policy in his model and he just tried to find the clusters which can has the better performance for giving the similarities among items.

2.10 Evaluation Methods

Mostly, there are three different types of RS evaluations; offline, user studies and online experiments. Often is easiest to perform offline experiments using available dataset. I used this method in this research as well. There are two classic tasks in recommender systems: rating prediction and item ranking (also called item recommendation, or top-N recommendation). Rating prediction is to predict the rating user u would give to item i. on the other hand, item ranking is to provide a user with a ranked list of items. The often used metrics for explicit feedback recommendation are error based metrics which can be used to measure the recommendation accuracy. This measures the distance between the predicted ratings and the ratings in the test set such as Root Mean Squared Error. For item ranking task, it is better to use list based measurements. This is because although items can be rated by their predicted ratings, error based metrics and list based metrics usually do not agree on the ordering of algorithms (for example, the best rating predictor might be the worst when it comes to ranking). Therefore, based on different method, this suggestion about different metrics can be applied [Li 2017, Shani 2011]:

- For explicit feedback based recommender systems, if the recommendation task is rating prediction, use error based metrics, and if the recommendation task is item ranking, list based metrics should be used;

- For implicit feedback based recommender systems, since explicit rating is not available and therefore the recommendation task can only be item ranking, list based metrics should be used.

2.10.1 Error based metrics

Error based metrics measure how close the ratings estimated by a recommender are to the true user ratings. The most important representatives are mean absolute error (MAE), mean squared error (MSE) and root mean squared error (RMSE).

2.10.2 List based metrics

List based metrics are used to evaluate top-N recommendations. Since in this research our model is a top-N recommendation task, I focus on these metrics. The evaluation metrics are calculated on a test set. The relevant and irrelevant items should be defined clearly based on the final goal of the recommendation systems. List based metrics include classification accuracy metrics and ranking accuracy metrics.

	Relevant	Irrelevant	Total
Recommended	tp	fp	$tp + fp$
Not Recommended	fn	tn	$fn + tn$
Total	$tp + fn$	$fp + tn$	N

Figure 2.4: Confusion Matrix [Li 2017]

2.10.2.1 Classification accuracy metrics

Classification accuracy metrics are calculated from the number of items that are either relevant or irrelevant and either contained in the recommended list or not. These numbers can be clearly arranged in a contingency table that is sometimes also called the confusion matrix (see Figure 2.4). The most commonly used classification accuracy metrics for evaluating top-N recommendations are as below.

1. **Precision**: The ratio of the recommended and relevant items to the recommended items.

$$precision = \frac{tp}{tp + fp} \tag{2.4}$$

2. **Recall**: The ratio of the recommended and relevant items to the relevant items.

$$recall = \frac{tp}{tp + fn} \tag{2.5}$$

3. **F1 score**: The harmonic mean of the precision and recall.

$$F1 = 2.\frac{precision.recall}{precision + recall} \tag{2.6}$$

4. **AUC (Area Under Curve)**: It is most commonly referred as area under ROC (Receiver Operating Characteristic) curve. This score for recommendation systems shows the probability that a randomly chosen positive example has a higher score than a randomly chosen negative example. The score lower than 0.5 means no better than a random model. In some cases, AUC corresponds to the area under the precision-recall curve from recommending 1 to N items, which is also referred to as AUPR (Area Under the Precision-Recall curve).

5. **MAP (Mean Average Precision)**: For each item in the test set, the precision at the position of every correct item in the ranked results list of the recommender has been calculated. The mean of these average precisions across all users is the mean average precision or MAP. At rank k in the predictions list MAP is defined as:

$$\text{MAP} = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{2.7}$$

where Q is the set of queries which the system will show some results based on that and AP is

$$\text{AP(q)} = \frac{\sum_{k=1}^{n}(P(k) \times rel(k))}{\text{number of relevant items}} \tag{2.8}$$

where p(k) is precision at top(k) and rel(k) is a binary function which defines if an item is relevant (1) or not (0).

However, we should be aware that the metrics which consider blank zero's value as false positive or irrelevant in implicit data are not always disliked items by users. Then, they may not be the best metrics, since it assumes that a user dislikes the items that are not in the testset but that is not reliable information [Steeg 2015].

2.10.2.2 Ranking accuracy metrics

Ranking accuracy metrics are calculated by taking into account the order of items in the ranking list. The most commonly used ranking accuracy metrics for evaluating top-N recommendations are as below:

1. **NDCG (Normalized Discounted Cumulative Gain)**: is a normalization of the Discounted Cumulative Gain (DCG) measure at rank n of the ideal ranking. The ideal ranking would first return the items with the highest relevance level, then the next highest relevance level, etc.

$$NDCG = \frac{\frac{1}{N}\sum_{u=1} N \sum_{j=1} J \frac{g_{ui_j}}{max(1, log_b^j)}}{DCG^*} \tag{2.9}$$

   Where DCG^* is the ideal DCG.

2. **MRR (Mean Reciprocal Rank)**: MRR is the average of the Reciprocal Rank (RR) across all the recommendation lists for individual users. RR measures 1/the rank of the highest ranked positive example. MRR can be computed as:

$$MRR = \frac{1}{|Q|}\sum_{i=1} |Q| \frac{1}{Rank_i} \tag{2.10}$$

   where Q indicates the relevant items in the test set for each user.

3. **WRank (Mean Weighted Rank)**: For each user,the preferences for all items will be predicted and they will be sorted to get an ordered list. Then, rank(i, j, k) denotes the (relative) position of item j in that ordered list, for user i and context k. Finally if we use this evaluation rank for our recommendation system, the ranks are weighted according to how many times a user (company) booked a hotel in specific City (context). That means that the rank for a hotel that the user booked very often in the test set is more important than a hotel the user booked only few times in each city [Steeg 2015][Hu 2008].

$$WRank = \frac{\sum_{i,j,k \in testset} r_{ijk} \times rank(i,j,k)}{\sum_{i,j,k \in testset} r_{ijk}} \tag{2.11}$$

   A Lower value is better. WRank of 0 is considered the best score, since that means that the right items (i.e. the hotels in the test set) are the ones with the highest prediction scores.

Chapter 3

Data Analysis and Exploration

The first and the most important step for solving a problem is understanding the problem completely. One of the methods for understanding a problem is through understanding the data. First, we need to preprocess data in order to have clean and usable data to understand it better. Here, there might be missing values, duplicated data, or noise which should be handled. In this chapter, firstly, I gave a general overview about the data which we have access to them but before that, I explained the preprocessing steps. The main goal of analysis and exploration of data in this step is to find all related and important features in booking data for the business policies. Also, these extracted features can be considered as observable business policies. I explain how we chose them in this step and finally how we want to use them in our recommendation system model.

3.1 Cleaning and Preprocessing Data

The available dataset was in csv format and includes 7 month of booking data through travel agencies which booked a hotel for business travelers from different companies. It includes 2946113 records with 18 features for each record.

3.1.1 Duplicate Values

There were some duplicate cases inside the dataset, I removed all the duplicates before applying next steps on data. The new dataset had 2,916,355 rows with 18 features.

3.1.2 Cleaning and Missing Values

The name of the companies are not defined in the booking data. Therefore, we should later find a strategy to find the name of the company associated to each transaction. When we tried to upload the csv file in a dataframe, we observed for some of the records, there are a 19th feature which is created by mistake through the manual entering of data to the system and for two records, there was an extra feature which we should handle. 9 features are categorical and 9 features are numerical attributes. one general overview about data can be seen in table 3.1.
First of all, in the cleaning process phase, we should find a way to add the name of the companies per each booking since they are defining the name of users in utility matrix of our recommendation systems. we didn't have the name of the companies, we just had the business email addresses of customers, we decided to extract the name of the company from the domain name of these email addresses.

Attribute	Description	Type	Distinct Value	Missing data
Date	The date of booking hotel	String	2405659	0
propertyId	Id of booked hotel	String	94314	0
chainCode	Id of the Hotel chain	String	319	0
rateCode	Id of negotiate policy	String	11455	0
roomType	Id of the Room Type	String	4679	0
leadTime	Number of the hotel booking days in advance	Integer	458	0
taSignIn	The Id of each travel agent	String	36367	0
rooms	Number of the booked rooms	Integer	2	67
nights	Number of the booked nights	Integer	98	0
guests	Number of guests per each booking	Integer	7	52356
adults	Number of adults per each booking	Integer	7	0
supplierCode	Id of the Supplier	String	3	2941578
totalPriceEUR	Total final price for each booking	Integer	259669	1730124
baseAmountEUR	initial price per each booking	Integer	333424	562074
comissionEUR	The commission of the travel agency	Integer	227769	1839625
markupEUR	no specific definition	Integer	1781	2942807
officeID	Id of travel agency	String	5911	0
email	Email address of business traveler	String	1044875	0

Table 3.1: general Overview

Second, We didn't have the name of the city in which the hotel was booked, the name of the city is so important since the recommendations should just include the hotels in a specific city as the destination city of the customer. Name of city obviously is the first and obligatory context which we should consider in our recommendation systems. Based on data, we had a property Id feature which identifies hotels. It includes a string with length of 8. Two first character presents chain code of the hotel, the next three character presents airport code (IATA airport codes[1]) and the last three characters represent hotel Id. We extracted the information related to the hotel chain code and airport code as the city code from this string.
Here, you can find the detailed of preprocessing steps for each attribute:

1. **Company name**
   As we already mentioned, there were some limitations about extracting the name of the company from the business email address:

   - In some records, any kind of email addresses are not provided,
   - In some of them, more than just one email address is provided,
   - In some records, customers did not provided business email addresses; just public domain emails,
   - In some records, travel agencies give their email addresses instead of business email addresses of customers

   If there is not any business email address in records or if there was an explanation such as a reference number to other documents to find the email addresses or even there was an ambiguous explanation about email addresses, I change the email addresses to null value. If extracted domain name of the emails include some of the

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

public domain names like gmail, yahoo, icloud, outlook and etc., I also changed the email addresses to null value. If there was more than one email address, mostly they were from the same company, then I just considered the domain name of first business email address.
For some other records, we just had the telephone number of the customer which was entered mistakenly instead of the email address. I first deleted the all digits from the string and after that I changed the value of all cells which include just special character like +,- and etc to null value.
Also, in some cases, there were some mistakes for entering @ symbol like @@ or @@@ and in some cases, the domain address of email were split from other parts by some different signs like "//" or "." . In some other records, @ Symbol were written in some other forms such as "AT" or ".AT." or " . ".
Also, we had access to the name of the travel agencies in a hierarchical structure. These names are not exactly the same as the name extracted from email domain addresses. we just around around 250 exact names and we just removed these names. Finally, for all of null value in email address attribute, I replaced them by the name "NoCompanyName". Around 5% of all dataset has the "NoCompanyName" instead of the real name of the company. There are some strategy which can be done to fulfill the missing value of the name of the company with real name or the name with high probability. I did not apply them in first place here, but I explained them later how we can fill them in the final chapter. The top 20 companies which had the most bookings in this dataset can be seen in figure 3.1

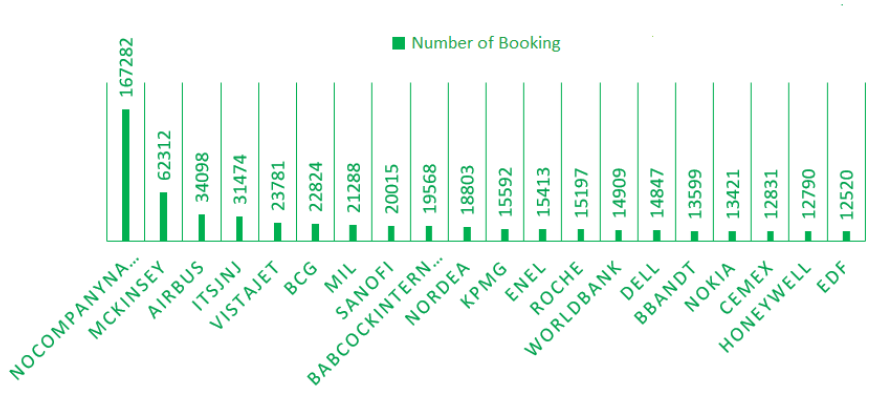


Figure 3.1: Top 20 companies with highest number of booking

2. **City Code**
City code can be extracted from property Id. This code is a three length character code embedded in property Id from character 3 to 6. Finally, we had 4105 unique codes for cities. This code is exactly the closest airport code of each city but for some cities which there are lots of request to book a hotel, if even there is not any specific airport for them, they have a unique city code such as hotels in Sophia Antipolis which based on airport code should have the code NCE (Nice airport as the closest airport), but the city code of their hotels are SXD. Some Cities can have

the same citycode since the closest airport to them is the same. The distribution of destination city code of different companies can be seen in figure 3.4. The color is based on the different companies. The city code by using a dictionary can be mapped to the real name of the city and also to the real name of the country. The top 20 city which has the highest booking number can be seen in figure 3.2 and the top 20 countries which has the highest number of booking can be seen in figure 3.3.

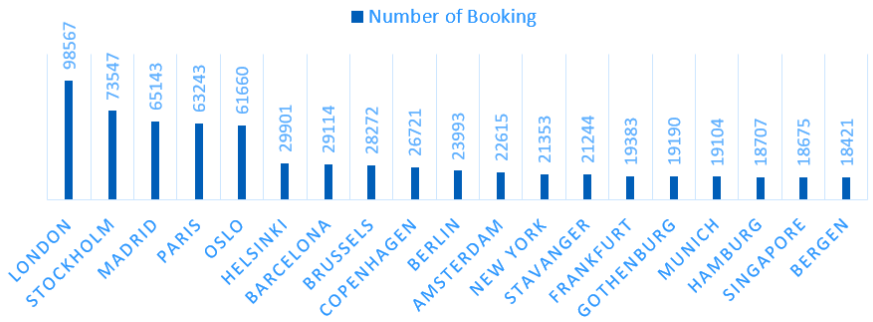


Figure 3.2: Top 20 cities with highest number of booking

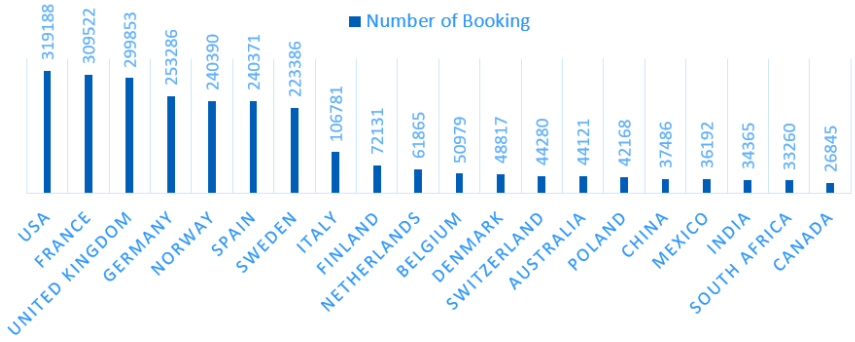


Figure 3.3: Top 20 countries with highest number of booking

3. **propertyId**
   We mapped the propertyId of hotels with real name of these properties from another database. In this case we will have a better sense about the items which we want to recommend. The name of top 20 hotels with highest number of booking can be seen in figure 3.5. Also, we mapped the ID with the location of that hotel with the city which they are located in. The unique values for these city names are 17018.

4. **Price**
   Total final price per each night for each room should be calculated by dividing total price by room multiplying by length of stay (number of nights). The value of nights for some rows is zero, which means the hotel room just was booked for a few hours, in this case the pricepernightroom was just considered as the price per each room.

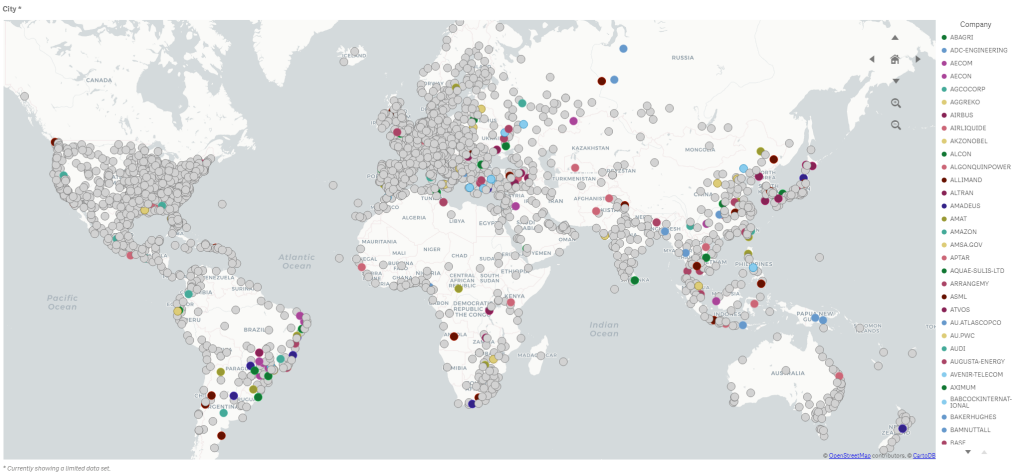


Figure 3.4: Distribution of city codes (airport location)

PROPERTYID	HOTEL NAME	CITY	COUNTRY	NUMBER of BOOKING
TFOSLOPE	THON HOTEL OPERA	OSLO	NORWAY	6463
RDOSL631	RADISSON BLU OSLO AIRPORT GARDERMOEN	GARDERMOEN	NORWAY	6074
SHSVG325	SCANDIC STAVANGER AIRPORT	STAVANGER	NORWAY	4978
CIBGO142	COMFORT HOTEL BERGEN AIRPORT	BLOMSTERDALEN	NORWAY	4286
CCARN104	CLARION HOTEL ARLANDA AIRPORT	STOCKHOLM	SWEDEN	4198
CCSTO081	CLARION HOTEL SIGN	STOCKHOLM	SWEDEN	4176
QISVG015	QUALITY HOTEL STAVANGER AIRPORT	SOLA	NORWAY	3918
RDOSL493	RADISSON BLU PLAZA HOTEL OSLO	OSLO	NORWAY	3832
CCBGO098	CLARION HOTEL BERGEN AIRPORT	BLOMSTERDALEN	NORWAY	3457
SHBGO782	SCANDIC FLESLAND AIRPORT	BLOMSTERDALEN	NORWAY	3425
RDCPHS02	RADISSON BLU SCANDINAVIA CPH	COPENHAGEN	DENMARK	3229
CCSVG128	CLARION HOTEL ENERGY	STAVANGER	NORWAY	3158
SHOSL766	SCANDIC BYPORTEN	OSLO	NORWAY	2960

Figure 3.5: Top 20 hotels with highest number of booking

5. **Guests**
   For all missing data of guests number, the value of adults column can be used. For all other data, just around 80 cases the number of guests are more than adults. Between guests and adults can just choose the adults column and remove the guests column. I just changed the name of the adults column to the guest column. The number of rooms for each case should not be greater than the number of adults which no case was found.

6. **Suppliercode**
   For all of missing data of supplier code, I just decided to write the 'DEF' as abbreviation of default. It's normal to have the supplier code for just a few specific chain hotels.

7. **totalPriceEUR**
   For all of the cases for which total price is missing, if the value of the base amount (baseAmountEUR) exists, I replaced the base amount value with missing total price values.

8. **leadTime**

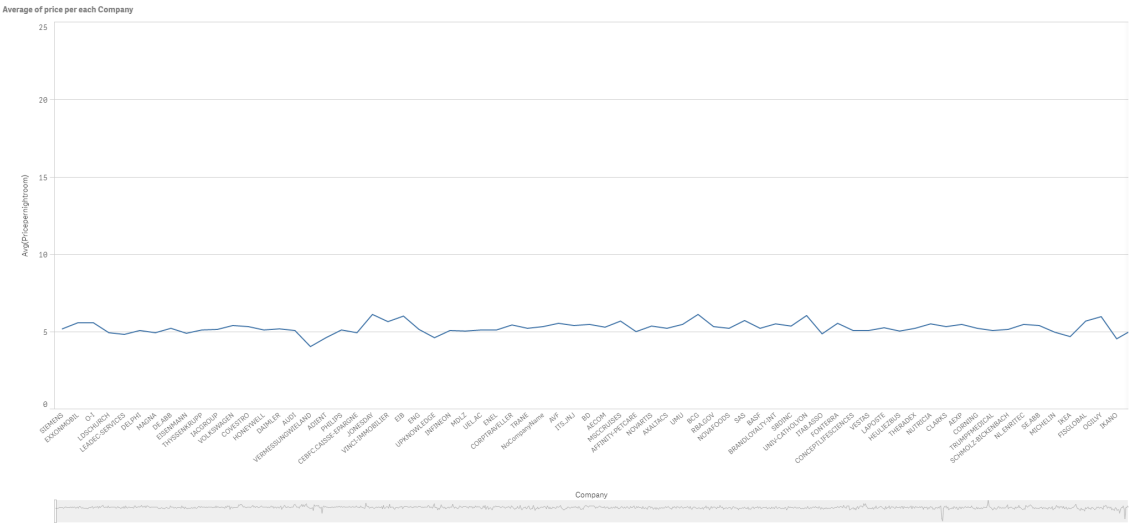


Figure 3.6: Average cube root pricepernightroom per each company

Negative lead time means the days which the customer canceled the booking before departure. I just decided to keep the booking which they did not cancel. I removed the 2784 rows from the dataset related to the canceled booking.

9. **Date**
   The date and the time of each transaction of the booking is available. As we already mentioned this data is just for 7 month from 01/09/2017 to 27/03/2018. We extracted the months of each transaction from this date. The distribution of all bookings based on different months is available in figure 3.7. Except on the month December, for which there is a long public holiday (Business traveling occurs less than other months), the number of bookings is distributed equally between different months.

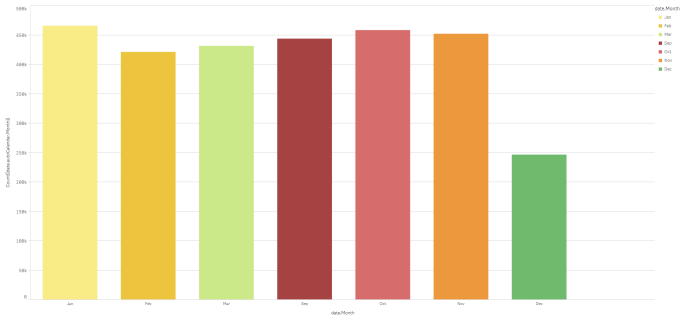


Figure 3.7: Distribution of historical transaction based on different months

10. **taSignIn**
    I removed this feature because it's just the Id of staff from each travel agency which signs in inside the system. It doesn't give us any helpful information.

11. **officeID**

Table 3.2: Basic statistics

Attribute	count	mean	std	min	25%	50%	75%	max
leadTime	2916353	15.21	23.11	1	3	8	19	370
rooms	2916353	0.98	0.13	0	1	1	1	1
nights	2916353	2.01	2.38	0	1	1	2	99
guests	2916353	1.02	0.15	1	1	1	1	8
totalPrice	2916353	266.35	445.96	0	107	163.14	301.37	335365.9
BaseAmount	2916353	225.2	427.59	0	79	138.45	270	335365.9
Comission	2916353	9.44	36.07	0	0	0	11.12	33536.59
Markup	2916353	0.03	1.63	0	0	0	0	864.2
Pricepernightroom	2916353	131.23	107.36	0	89.09	118	156.87	84140.65
Cube root[2]	2916353	5.385	1.103	0	5	5	6	44

By considering travel agency code (officeID), part of characters inside the code seem to be repeating in different codes. It became clear that characters 3 to 5 inside this ID is representative of the travel networks of a group of travel agencies. I extracted them as a separated column called TravelNet.

3.1.3 Transforming Data

Based on the basic statistics for numerical attributes in table 3.2, we can see the distribution of the all of numeric data, except rooms and guests, has a strong skewness. However, this skewness is stronger for totalprice values. Usually log transform and Cube root transform work well in transforming data to the normal distribution. I did both of them and because cube root decreased the range of price data more, I just chose the cube root transform for totalprice data. The last raw of the table 3.2 is the basic statistic of price per night and room after applying cube root transform. A part of average price of each company is shown in figure 3.6.

1. **Pricepernightroom**
   I replaced the ceiling value of cube root transform for totalprice data with original value.

2. **comissionEUR**
   I replaced the ceiling value of cube root transform for comissionEUR data with original value, then we have categories of commissionEUR.

3. **leadTime**
   I did the same transform process with leadTime data to replace the ceiling value of cube root by original value.

Based on the variance of numerical data, the variance of room and guests columns are too low and also we considered their values in calculating the priceperroomnight; Since the columns with low variance cannot have any specific effect on finding any similarity between records, I just decided to remove room and guest columns.
Also, I saw the distribution of average price which each company paid for booking the hotels monthly 3.8. The average of price are mostly similar for different months for each company but I noticed that some companies don't have any booking in some months then I just decided to add month column, because it helps to find the companies which have the

Table 3.3: Unique values and types of final attributes

Attribute	Unique Values	Type
Date	2396781	String
propertyId	94074	String
hotelName	94074	String
chainCode	319	String
rateCode	11445	String
roomType	4676	String
nights	98	String
leadTime	361	Integer
supplierCode	3	String
comissionEUR	23	Integer
officeID	5870	String
TravelNet	1356	String
Pricepernightroom	27	Integer
company	38308	String
cityCode	4106	String
cityName	17018	string
countryCode	202	String
Month	7	String
Year	2	Integer

similar business needs (they need business traveling all the time like consultant companies or just they need it in some specific months).

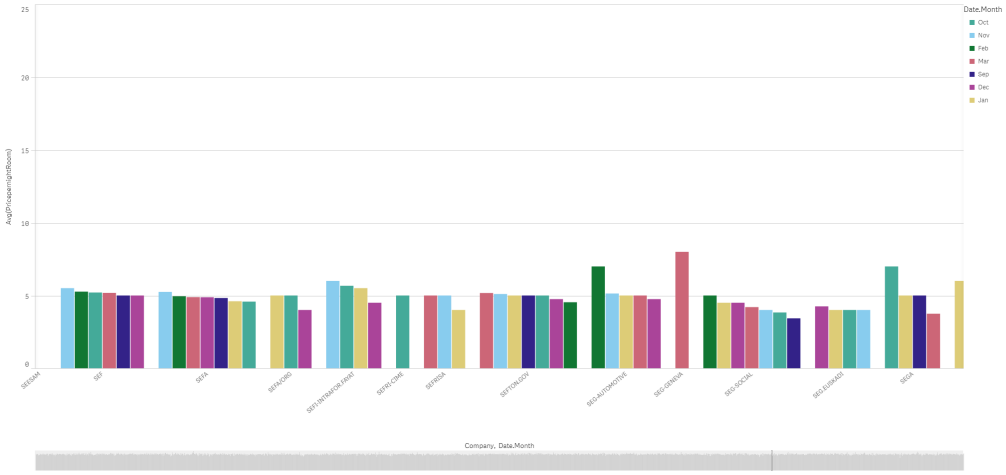


Figure 3.8: Averages prices for each company in different months

The final attributes and their unique values of them is available in table 3.3.

3.2 Inferring Business Policy from Dataset

The most important question for our problem is what are the most important features in our dataset which can define the business policy. Amadeus doesn't know about the business policy. They just know that each company have contracts with some travel agencies and those travel agencies has contracts with some hotels, but the detail of theses business policies and what is mentioned inside the contracts is not known by Amadeus. One method which can apply here is analyzing data to see correlation and occurrence of

attributes to each other to infer the business policy. The method is like feature selection to find the most important features like the business policy. Feature selection is a common technique used in data preprocessing for pattern recognition, machine learning and data mining [Xie 2009]. Using the chain code or property Id code as the final label to consider supervised machine learning algorithm was the idea which was done in the master thesis project in last year in Amadeus. Since the cardinality of most of features are high, it doesn't seem a good way. Also, by clustering the hotels, we cannot extract the important features related to business policy. Business policy should consider three entities of hotels, companies and travel agencies and applying supervised clustering method on any of these entities ignores the importance of other entities. Then, we should use unsupervised feature selection. Another limitation is about type of data in our dataset which is mixed between numerical and categorical; However 14 features have categorical value and even 4 features with numerical value can be interpreted as the category. First maybe using some methods to represent the data in numerical ways such as one hot encoder seems one option; however, the cardinality of most of features are too high, then using encoder to represent the categorical value to numerical value doesn't seems practical. Thus, I should find methods for applying unsupervised feature selection on categorical data. Frequent itemset mining and association rules mining seems best algorithms for this task. Since the diversity of most of features in dataset is very high, statistical methods such as Chi square cannot apply here as well.

3.2.1 Frequent Itemset Mining

Frequent itemset mining is one of the most well known techniques to extract knowledge from data. This method has been an important part of data analysis and data mining. Frequent itemset mining tries to extract information from dataset based on frequently occurring events, and is interesting if it occurs frequently in the data, according to a user given minimum frequency threshold. The main restriction of this method is combinational explosion of results for a big dataset. It needs lots of memory for extracting these itemsets. One solution we thought about was just extracting the patterns which we assumed to be correlated to each other and check they are frequent or not. We just tested the frequency for just 54 different patterns with combination of 8 different attributes. Part of these patterns can be seen in table 3.4.

The part of the result for pattern 4, can be seen in table 3.5. I considered a threshold of 25% for the occurrence of more frequent itemset in dataset which it means over 750000. The threshold for considering an itemset as a more frequent itemset is at least 10. The number of occurrences of these parameters with each other like the frequent itemset in our dataset can be seen in table 3.4. I explored that for all 54 patterns, there are lots of frequent itemset as the result from our big dataset, then we can assume,these 8 features are correlated to each other. List of these features are as followings, they are selected based on the Amadeus experts' opinion and business rules: *company name, rate code, chain code, city code, roomType, officeID, Suppliercode, Totalpricepernight*

Table 3.4: Frequent patterns

Pattern	Occurrence	Diversity of Itemset
company , officeID , rateCode	2316996	31485
officeID , rateCode	2693467	21946
company , rateCode	2491862	25061
company, officeID , rateCode, ChainCode	1700277	39964
ChainCode, officeID , rateCode	2173164	43342
company, officeID , rateCode , city , PropertyID , RoomType	726077	28193

Table 3.5: Frequent itemset in one pattern

Pattern	Frequency
MCKINSEY|MUCAP21WH|MCK|ZQ	9002
NoCompanyNames|PARBL4100|RES|BL	7294
EDF|EBUWL2200|EDF|RT	5482
BABCOCKINTERNATIONAL|MANFB3603|71B|HI	4783
NORDEA|CPHBA28CG|NRD|RD	4278

3.2.2 Association Rules Mining

Association Rules Mining is a method that is useful for discovering interesting relationships hidden in large data set. It was initially used to market basket data for finding relationships existing among the products which was useful for cross-selling their products to customers. It was a solution for combinational explosion of results of frequent itemset mining method. For having an overview about this method, first we need to know the following related terms [Rajeswari 2015, Shahzad 2013, Xie 2009]:

1. Itemset and Support Count
   In association rules mining, first we should know the definition of itemset which is a collection of zero or more items. An itemset which contains k items is called a k-itemset. An important feature of an itemset is its support count, which refers to the number of instances that contain a specific itemset.

2. Support and Confidence
   An association rule is an indication term of the form $X \rightarrow Y$, where X and Y are disjoint itemsets. The importance of an association rule is measured by its support and confidence. Support figures out how often a rule is frequent in a given data set, while confidence figures out how frequently items include Y appear in itemset that contains X.

3. Lift
   Lift is a correlation measure which is used to extend the support and confidence for association rules. If the occurrence of itemset A is independent of the occurrence of itemset B, we know $P(A \cup B) = P(A)P(B)$; Otherwise, itemsets A and B are dependent and correlated as events. Also, this definition can easily be extended to more than two itemsets.

   The lift(A, B) less than 1 means then the occurrence of A is negatively correlated with the occurrence of B. The lift(A, B) greater than 1 means that A and B are positively correlated which defines that the occurrence of one infer the occurrence of the other. If the lift(A, B) is equal to 1, then A and B are independent and

there is no correlation between them. The value greater than 1, shows the positive correlation and when the value is bigger, it shows stronger correlation between A and B.

4. Apriori algorithm
   The first association rules mining algorithm which is pioneered the use of support based pruning to systematically control the exponential growth of candidate item sets is Apriori algorithm. The procedure of association rules mining in this algorithm is divided into two steps: (1) The first step is to iteratively find out all frequent itemsets whose supports are not less than the user-defined threshold. (2) The second step is to construct association rules that satisfy the user defined minimum confidence by using frequent itemsets. Then other rules are generated by deleting the last items in the antecedent and inserting it to the consequence, further the confidences of the new rules are checked to determine the interestingness of them. Those processes iterated until the antecedent becomes empty [Kotsiantis 2006].

In first step, I ran the apriori algorithm for the whole dataset with min support = 0.0008 and min confidence = 0.2 and min lift =3 for defined minimum length of 2, because we are searching to find at least the correlation between two parameters. we found just 1382 rules which they were fit on our conditions. In the Second step, I ran the algorithm for each company having more than 1000 different bookings, we tried to find the most correlated features based on the different companies. I chose the min-support of at least happening 100 times per all of the transaction of the company which is at least equal to 0.1. In general, we had 464 different companies which I chose min-lift of 1 for finding all of correlation. Based on the two steps, the final result infer that 8 variables are strongly related to each other, since the longest itemset can be mapped to these 8 features:
company name, rate code, chain code, city code, roomType, officeID, Suppliercode, Totalpricepernight

One example of our result is as following:

items=frozenset('DEF', 'ROH', 'MILAX21BN', 'HS', 'FAT', 'FCAGROUP','5'), support=0.07995707002951435, confidence=1.0, lift=4.781270044900578

Implementation

For Implementation the apriori algorithm, I used mlxtend Python library which is open source and can be found in Github[3]. The most frequent itemsets are returned in form of frozenset, which is built-in Python type that is similar to a Python set but immutable, which makes it more efficient for certain query or comparison operations. Then the order of item in each itemsets is not important. I used m5.24xlarge cpu instance from AWS with 384 GB memory for applying apriori algorithm.

[3] https://github.com/rasbt/mlxtend/

3.2.2.1 Known Business Policy

Finally based on our result in this section, we can conclude there is a strong relation between 8 variables:

company name, rate code, chain code, city code, roomType, officeID, Suppliercode, Totalpricepernight

So far, we can conclude that the core of our business policy is built based on these features. But It does not mean that we can ignore other features. Also, maybe there are other latent and unknown features which have an important effect on business policy but they are not visible for us, then we should find a way to extract them. I explain more about these latent features in chapter 5.

To explore the data easily, I made a dashboard through Qlik sense to show the dependency of these features to each other.

3.3 Collecting Implicit Feedback from Users

Gathering Implicit feedback from users to create the utility matrix can be done in different way. We can just consider the binary preferences of booking or not booking hotels in the past, or we can consider the number of the bookings of each hotel by each user in the past and normalize or scale the values.

3.3.1 Utility Matrix

Utility Matrix forms the basis of any recommender system. As mentioned before, there are two ways to fill the utility matrix data: with explicit rating from users; or with implicit ratings inferred from observing user behavior. The dimensions of utility matrix is usually in a 2D aspect, $Users \times Items$. However, If we consider context, context can be considered as third dimension and combine a set of 2D matrices to create a 3D matrix dimension of $Users \times Items \times Context$. For our problem, we want to recommend hotels to business travelers (companies), then from booking data we can implicitly extract the number of times a company booked a hotel as the 'rating' value of the utility matrix $Companie \times hotels$. In this way, we have lots of unknown values since our utility matrix is so sparse (see table 3.6).

If we choose considering context such as city, price, time and etc, based on high cardinality of each of these parameters as context we have a much sparser utility matrix, then we decided even for using context_aware recommendation system for our problem, it's better to use a post-filtering technique not considering the context to have a 3D utility matrix.

Another way is to use OLAP model. This model is similar to use pre-filtering but by aggregating data on different dimensions. In this case, still our utility matrix has two dimensions, but sometimes based on choosing aggregation, the filtered utility matrix is so small or void; then we cannot have recommendations. Having a utility matrix based on generalizing dimensions on coarse-grained level can be a solution.

Booking info	Number of users	Number of Items	Density
Starting	38308	94074	0.025%
Ending	13385	31657	0.178%

Table 3.6: Sparsity of the dataset

The main utility matrix is the $companies \times hotels$ matrix which based on our historical booking data includes 3,603,880,866 cells, and only 900,812 of them has the booking value. Sparsity always is a problem in real world can make the data messy and give inaccurate result.

If we consider the whole number of users and Items, we have the density of 0.025% which is too low (0.025% of utility matrix space is filled). In this case, even thinking about applying collaborative filtering techniques which are based on finding similarity between items or users cannot have a good result. For decreasing the sparsity, I decided to consider the users which made the booking more than 5 times and the hotels which booked more than 5 times to have a warm start to build a recommendation system. In this case, we have the density of 0.178% which is not that much better but is less sparse. Collaborative filtering still cannot have a good result in this case, then we should think about other methods.

Figure 3.9: an overview of built dashboard to show the dependency among observable features of business policies

Chapter 4

Integrating OLAP and Recommender Systems

Based on the criteria which we extracted from PNR in chapter 3 to understand what is the business policy, we can consider two complete different approaches. Assuming that our business policy is completely known for us which means that contextual factors are fully observable and static, we can model our problem by a multidimensional model based on OLAP.

Online Analytical Processing (OLAP) and data warehousing technologies have been popular and deployed in lots of businesses. The first and main idea for having a recommendation system for this problem by considering all of different extracted context as business policy from PNR data is to apply the powerful OLAP engines to speed up recommendation computation against large amounts of data. Amadeus was not sure that they want to build a data warehouse for this RS, then I just took the idea, used integration functionalities of OLAP and RS. In particular, I proposed such a prototype to aggregate and rollup hierarchical rating data, and incorporate traditional RS algorithms such as collaborative filtering at different levels [Fu 2016, Krohn-Grimberghe 2010, Aligon 2013].

Multidimensional Model

Multidimensional model is using not only on $Companies \times Hotels$ dimensions, as the classical (2D) dimensions, but also over several dimensions. If we consider our problem as a context_aware recommendation system with fixed and completely known context, which is extracted from PNR as our business policy in chapter 3, we can have multi dimensional data model used for data warehousing and OLAP applications in databases. We can consider each of the main important features of one transactions as different dimensions, such as *Companies, Hotels, Travel agencies,Time,Place*, and so on. If we consider $D_1, D_2, D_3, ..., D_n$ be dimensions, each dimension D_i is a subset of a Cartesian product of some attributes $A_{ij}, j = 1, 2, , ..., Ki$, where each attribute defines a value or domain; for example, Hotel dimension can be defined as:

$$Hotel \subseteq HName \times City \times ChainCode \times Price \times RoomType.$$

Initially, I just considered 6 different dimensions and I considered all important extracted features from PNR inside this dimension as their attributes:

Companies, Hotels, Travel Agencies, Price Category, Time, Cities.

The initial fact dimensional model of available data can be shown in figure 4.1.

Each Dimension can be defined as following:

1. $Company \subseteq CNames \times RateCodes \times Price \times OfficeIds$
2. $Hotel \subseteq PropertyID \times City \times ChainCode \times Price \times RoomType$
3. Travel Agency $\subseteq$ Travel Network $\times\ OfficeID \times Ratecodes \times Companies \times Hotels$
4. Price Category $\subseteq$ Price Category
5. $Time \subseteq Year \times Month \times Day$
6. $City \subseteq CityCode \times Country$

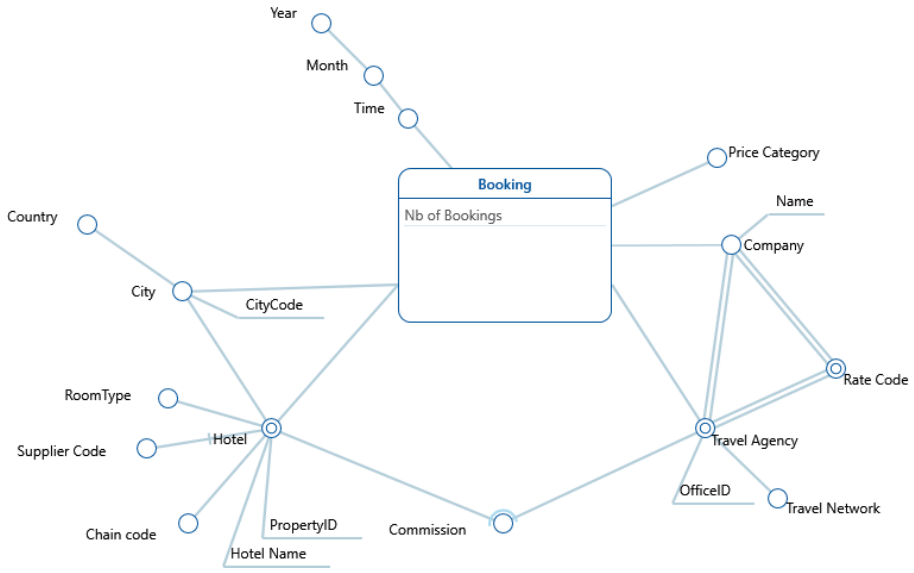


Figure 4.1: Fact dimensional model

Given dimensions $D_1, D_2, ..., D_n$, we define the recommendation space for these dimensions as a Cartesian product $S = D_1 \times D_2 \times ... \times D_n$. Moreover, let Rating be a rating domain representing the ordered set of all possible rating values. Then the rating function is defined over the space $D_1 \times D_2 \times ... \times D_n$ as

$$R : D_1 \times D_2 \times ... \times D_n \rightarrow Rating \tag{4.1}$$

For Example, considering the three-dimensional recommendation space $User \times Item \times Location$, we can define a rating function R on the recommendation space $Company \times Hotel \times City$ specifying how much Company traveler u ∈ User liked a hotel i ∈ Item at city c ∈ City, R(u, i, c).

The hierarchical structure of the dimension is shown in figure 4.2.

Aggregation based on this dimensions can result to gain a smaller utility matrix. We

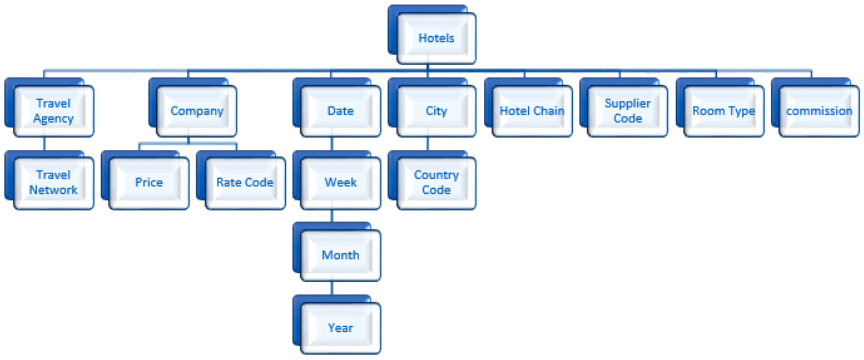


Figure 4.2: Dimension Hierarchical Structure

face two strategies here:

1. Using popular ranking : Proposing top N hotels per each company by using cluster based methods (grouping by methods here);

2. Using similarity based method by a generalized grouping by on a coarse-grained level.

4.1 Cluster based popular ranking method

In the first method, since groupby can even aggregate more fine-grained levels and our data is not provided hotels for all fine-grained level since our dataset is so sparse, then we should consider alternative grouping by method in aggregation on fine-grained level dimensions. In this case, one idea can be providing a weight for each dimension based on their importance on our business rules. This weight should be presented the priority of each dimension in the same level. For example if we did a grouping by based on the hotel chain, room type, month and city for a specific company and the result is null, city has the more weight than others in the first level, for other dimensions the priority is in this order based on business rule: hotel chain, date, commission, room type and supplier code, then we can redo the aggregation based on new grouping by on city and hotel chain - I chose just two dimensions not more, since the sparsity of dataset is high, the aggregation on less dimensions will give higher possibility to have final list of hotels as the result of aggregation.

If the result is still null, I just considered the dimension city and if still the results are null, I just recommend the top N hotels for all companies in that specific city. The pseudocode of this cluster based popular ranking is written in Algorithm 1.

Algorithm 1 Cluster based popular ranking algorithm

RQUIRE: The priority of each dimensions and hierarchical structure of them. City and Company should be part of Dimensions

INPUT: `Dimensions`, a list contains the value of dimensions for aggregation . `K`, the number of hotels which we want to recommend

OUTPUT: `ListofHotels`, returns a list of most booked hotel in those context

```
1: ListofHotels ← []
2: DictofHotels ← []
3: while DictofHotels!= [] do
4:   DictofHotels ← [SELECT hotels, Count(*) groupby Dimensions]
5:   if DictofHotels == [] then
6:     if length(Dimensions) > 4 then
7:       Dimensions ← two dimensions with highest priority from Dimensions + [City,
         Company]
8:     else
9:       Dimensions ← [City, Company]
10:  else
11:    Dimensions ← [City]
12: DictofHotels ← Sort DictofHotels based on their Count
13: ListofHotels ← [SELECT Top K hotel from DictofHotels]
    {If number of Input dimensions in addition of city and company are greater than 4,
    we choose two other dimensions except city and company}
```

I implemented this algorithm in Python by considering the weight and priority of all dimensions from business rules aspect. The prototype gives a list of propertyIds and their names sorted based on the frequency of times which is booked by considering the aggregation of the different dimensions. The result of this prototype for company Amadeus, rate code AMA and city code PAR can be seen in figure 4.3.

```
RTPARMOU , IBIS ISSY LES MOULINEAUX : 5
CPPAR32D , CROWNE PLAZA PLACE DEREPUBLIQU : 5
UIPAR642 , CLASSICS HOTEL PORTE DE VERSAILLES : 4
HIPARC12 , HOLIDAY INN PARIS-NOTRE DAME : 4
HIPARB23 , HOLIDAY INN GARE DE L EST : 3
```

Figure 4.3: Top 5 recommended hotels for company 'AMADEUS' with ratecode 'AMA' in Paris

If we search for a completely new company, the result as you can see in figure 4.4 will be the top popular hotels on that city.

```
RTPARDFS , MERCURE PARIS LA DEFENSE : 1901
SMPARDEF , MELIA PARIS LA DEFENSE : 1705
MDPARMER , LE MERIDIEN ETOILE : 1386
RTPARORL , NOVOTEL PARIS PORTE D'ORLEANS : 1211
RTPARORG , NOVOTEL POISSY ORGEVAL : 1095
RTPARLDF , NOVOTEL PARIS LA DEFENSE : 1080
RTPARARM , MERCURE PARIS GARE DE LYON TGV : 1026
RTPARLAD , IBIS PARIS LA DEFENSE CENTRE : 1002
BLPARP11 , BALLADINS ESBLY - MARNE-LA-VALLEE : 948
BLPARP12 , BALLADINS GENNEVILLIERS : 909
RTPAREXP , IBIS PARIS BERCY VILLAGE : 903
RTPARPLA , MERCURE PARIS PTE VERSAIL EXPO : 891
RTPARMTT , IBIS PARIS MONTMARTRE : 846
WIPAR729 , THE WESTIN PARIS VENDOME : 780
RTPARMTR , MERCURE MONTMARTRE SACRE COEUR : 756
AZPARDEF , CITADINES LA DEFENSE PARIS : 754
BLPARP09 , BALLADINS COIGNIERES : 740
PUPARARC , PULLMAN PARIS LA DEFENSE HOTEL : 691
RTPARMRG , MERCURE PARIS PTE D'ORLEANS : 679
RTPARISS , NOVOTEL SUITES PARIS ISSY : 677
RTPARLYO , NOVOTEL PARIS GARE DE LYON : 640
```

Figure 4.4: Top 20 recommended hotels for a complete new company 'SAMANU' in Paris

4.2 Similarity based method

In this method, we firstly wanted to consider a fixed group of dimensions per different queries. The combination of travel network keeps all of the companies which booked hotel from the same travel network agency and city keeps all hotels which were booked by different companies in the same city. The problem is because of high cardinality of travel agency and city dimension, based on different queries, we should keep the aggregation result of 1356 different travel networks and 4106 different city codes. It's kind of like pre-filtering on coarse-grained level dimensions. Since It was taking too much time for running this model for whole combination of $TravelNet \times CityCode$, I just considered a few combination just for testing this idea. Since my final utility matrix extracted from this pre-filtering is not sparse anymore (sparsity for some random aggregation of these

dimensions were around 3% to 10%), I can use similarity based collaborative filtering method such as K nearest neighbors (KNN). I chose KNN similarity based model, and dot product similarity based such as SVDFUNK and SVD++ model.

I considered one hotel per each company in evaluation dataset and one in test set, and remaining hotel was considered for training dataset. Thus, the number of booked hotel per each company should be greater than 3. I just considered the companies and the hotels with booking times more than 3 in the dataset. Another difficulty for this model is finding hyperparameters which should be done per each aggregation on these two dimensions separately, since the data, the size of utility matrix and the sparsity of dataset is completely different.

SVDFunk

This method is part of matrix factorization based method. However, since I did not use it as the final model for the whole dataset, I did not explain it in next chapter. I introduce briefly this model here.
The SVD algorithm (Singular value decomposition) have become very popular after applying by Simon Funk during the Netflix Prize competition. The prediction $\hat{r}_{u_i}$ is set as:

$$\hat{r}_{u_i} = \mu + bu + bi + q_i^T p_u \tag{4.2}$$

If user u is unknown, then the bias bu and the factors pu are assumed to be zero. The same applies for item i with bi and qi. When baselines are not used, this model is equal to probabilistic matrix factorization (PMF). The parameters are learned using a SGD method on the regularized squared error objective. For details, see equation 5.3. In figure 4.5, part of the prediction result can be seen for all of the hotels in city code **OSL** per each company in same travel network **LBE** in the test dataset. This model is not designed for implicit data, then I scaled the range of booking times for hotels to a range between 0.5 to 5. The RMSE for the model for train dataset is equal to 0.07033, in this process also I found the hyperparameters based on GridsearchCV in this method for this specific aggregation of dimensions, hyperparameters are as following:
'n_factors': 100, 'n_epochs': 100, 'lr_all': 0.005, 'reg_all': 0.4
where lr_all is the learning rate and reg_all is the regularization term.
The RMSE for test dataset is equal to 0.1325 which means model predicted the number of bookings with high accuracy.
The search space is as follows:
param_grid = 'n_factors':[10, 200],'n_epochs': [5, 100], 'lr_all': [0.002, 0.005],'reg_all': [0.1, 0.6]

SVD++

The SVD++ algorithm is an extension of SVD method taking into account implicit ratings. Here, an implicit rating describes the fact that a user u rated an item j, regardless of the rating value. The search space is like SVD model and hyperparameters are as following: 'n_factors': 10, 'n_epochs': 100, 'lr_all': 0.005, 'reg_all': 0.6.
The RMSE for the evaluation dataset is 0.00016, and for the test dataset is 0.00019.
The figure 4.6 shows part of the prediction of this model.

	uid	iid	rui	est	details	err
0	HYDRO	RDOSLS05	0.511790	0.503884	{'was_impossible': False}	0.007906
1	NRK	TFOSLAST	0.558952	0.553262	{'was_impossible': False}	0.005690
2	TROMSOHAVN	BXOSLGUL	0.500000	0.500000	{'was_impossible': False}	0.000000
3	BANENOR	TFOSLGYL	0.503930	0.573787	{'was_impossible': False}	0.069857
4	FKF	TFOSLTER	0.503930	0.514557	{'was_impossible': False}	0.010627

Figure 4.5: Part of prediction for test dataset based on SVDFunk model

	uid	iid	rui	est	details	err
0	ECP	TFOSLSPE	1.0	1.000158	{'was_impossible': False}	0.000158
1	ALESUNDKOMMUNE	CCOSL060	1.0	1.000003	{'was_impossible': False}	0.000003
2	STATKRAFT	TFOSLFJO	1.0	1.000018	{'was_impossible': False}	0.000018
3	DIFI	SHOSL391	1.0	1.000005	{'was_impossible': False}	0.000005
4	ANTICIMEX	CIOSL104	1.0	1.000001	{'was_impossible': False}	0.000001

Figure 4.6: Part of prediction for test dataset based on SVD++ model

KNN

KNN (K nearest neighbors) is a basic collaborative filtering algorithm. I used cosine and Pearson as my similarity measure. The final evaluation result for test set can be seen in table 4.1. Also, part of prediction result based on cosine similarity can be seen in figure 4.7. The max number of neighbors taking into account for aggregation is considered 40 and The minimum number of neighbors is considered as 1.

	uid	iid	rui	est	details	err
0	KRISTIANSANDKOMMUNE	SHOSL390	0.500000	0.506680	{'actual_k': 40, 'was_impossible': False}	0.006680
1	LR	TFOSLFJO	0.500000	0.562753	{'actual_k': 30, 'was_impossible': False}	0.062753
2	UNI	TFOSLMUN	0.500000	0.509531	{'actual_k': 40, 'was_impossible': False}	0.009531
3	STAVANGERKOMMUNE	SHOSL334	0.515721	0.504127	{'actual_k': 40, 'was_impossible': False}	0.011594
4	MARINEHARVEST	SHOSL334	0.503930	0.503832	{'actual_k': 40, 'was_impossible': False}	0.000098

Figure 4.7: Part of prediction for test dataset based on KNN model-Cosine similarity

.

Comparing the result between SVD++ algorithm with other method is not correct since the dataset has binarized as input for SVD++ algorithm. However, if we give the scaled input to SVD++ algorithm, it will perform exactly like SVD algorithm. The final result to compare the performance of these models can be seen in table 4.1. As I already mentioned, this model takes lots of costs from the company aspect since we should have lots of different model because of high cardinality of dimensions. Then, another idea is just considering the original utility matrix, fill in the missing values, and after running the model, consider the context related to the users and items to do the post-filtering. I explain this method in next chapter 5. However, if we want to choose a model for just this specific filtered utility matrix, it seems SVDFunk has a better performance than KNN based similarity methods.

Method	Similarity Measure	RMSE	MAE
SVDFunk	dot product	0.0541	0.0178
SVD++	dot product	0.0002	0.0001
KNN	Cosine	0.0589	0.0203
KNN	Pearson	0.0708	0.0291

Table 4.1: Evaluation for different Experiment Models for just one specific generalized aggregation

4.2.0.1 Implementation

I used Surprise library which is a Python Scikit building and analyzing recommender systems [Hug 2017]. This library provides various prediction algorithms such as baseline algorithms, neighborhood methods, matrix factorization-based (SVD, PMF, SVD++, NMF), and etc. The code for this library is open source and can be found in the Github [1].

[1] https://github.com/NicolasHug/Surprise

Chapter 5

Matrix Factorization Based Models

The density percentage of the utility matrix for *companies* × *hotels* is 0.02% which is too low. Most of the similarity filtering methods do not have good result when sparsity is high. For solving the problem of the sparsity, another way is personalizing information. Matrix factorization that estimates the user liking for an item by taking an inner product of the latent features of users and items have been widely studied because of its better accuracy and scalability. Matrix factorization based models have been proven to be the most accurate and scalable for many cases [Krishna 2017, Koren 2015]. A basic matrix factorization model maps both users and items to their latent factor space of lower dimension. Accordingly, user-item interaction is modeled as inner products of the related user-item feature in that space. Thus, for vector q_i associated with each item i and vector p_u associated with each user u, the vectors define the extent to which an item described by those factors and the extent of interest the user has for items on those factors. The estimated rating r_{ui} for a given user u and item i is given by the dot product:

$$\hat{r}_{ui} = p_u.q_i^T \tag{5.1}$$

we do not know what these factors are. Nor do we know how many features (k) are relevant to describe an item and the interest of a user. We pick a number for k and learn the relevant values for all the features related to all the users and items. We will do the learning process by minimizing a loss function.

$$\hat{r}_{ui} = \sum_k p_{uk}.q_{ki} \tag{5.2}$$

where $\hat{r}_{ui}$ represents our prediction for the true rating r_{ui}, and $p_u.q_i^T$ is assumed to be a column vector. These user and item vectors are often known as latent vectors or low-dimensional embeddings. The goal here is to minimize the square of the difference between real number of bookings in our dataset and our predictions. This produces a loss function of the form

$$L = \sum_{u,i \in S} (r_{ui} - p_u \cdot q_i^T)^2 + \lambda_p \sum_u ||p_u||^2 + \lambda_q \sum_i ||q_i||^2 \tag{5.3}$$

Two L2 regularization terms at the end are added to prevent over fitting of the user and item vectors in formula 5.3. The goal now is to minimize this loss function. Derivative-based methods are methods for minimizing functions. I explain some of them in following:

5.1 Optimization Methods

Optimization is the process of finding the set of parameters which minimize the loss function. One method which is proven mathematically is we can find the best set of parameters

if we follow the gradient of the loss function. The gradient is just a vector of slopes (more commonly referred to derivatives) for each dimension in the input space. Gradient descent is one of the most popular algorithms to perform optimization. Calculation of numerical gradient is very simple using the finite difference approximation, but it is very expensive to compute. The second way to compute the gradient is analytically using calculus, which allows to derive a direct formula for the gradient (no approximations) that is also very fast to compute. The procedure of repeatedly evaluating the gradient and then performing a parameter update is called Gradient Descent.

5.1.1 Bayesian Personalized Ranking (BPR)

Rendle and et al. [Rendle 2009] presented a generic optimization criterion (BPR-Opt) for personalized ranking that is the maximum posterior estimator derived from a Bayesian analysis of the problem, also they provided a generic learning algorithm for optimizing models with respect to BPR-Opt using the likelihood function for $p(i > uj|\theta)$ and the prior probability for the model parameter $p(\theta)$. The probability of the user preferring the item over the other is the function of the difference of the prediction scores of the two items. Logistic sigmoid is used in such individual probability function.The learning method is based on stochastic gradient descent (SGD) with bootstrap sampling. The main idea here is based on sampling positive and negative items and running pairwise comparisons. It's necessary to know that in implicit feedback systems, only positive observations are available. The non-observed user-item pairs - e.g. a company has not booked a hotel yet - can be interpreted in two different ways and it is not clear which one is true:

- They are real negative feedback _ the business travelers from a company are not interested in booking rooms in the hotel
- They are missing values _ the business travelers from a company might want to book rooms in the hotel in future.

BPR in a simple way proceeds as follows:

- Randomly select a user u and then select a random item i which the user has booked it already. This is the positive item.
- Randomly select an item j which the user has booked on fewer times than item i (this includes even items that they have never booked). This is the negative item.
- Apply the standard collaborative filtering model which predict the preference, p_{ui}, for user u on positive item i. For matrix factorization, this may be $x_u \cdot y_i$.
- Predict the preference for user u and negative item j, p_{uj}.
- Find the difference between the positive and negative preferences $p_{uij} = p_{ui} - p_{uj}$.
- Pass this difference through a sigmoid and use it as a weighting for updating all of the model parameters via stochastic gradient descent (SGD).

- Notably, we do not care about the actual value of the preferences that we are predicting. All we care about is that we rank items which the user has booked on more frequently higher than items which the user has booked on fewer times.

Because this model use a sampling-based approach, it can be quite fast and scalable. Also since this method is sampling a positive and negative item for a user, predicting for both, and taking the difference, is using a pairwise strategy. Additionally, BPR directly optimizes the area under the ROC curve (AUC) which could be a desirable characteristic. Most importantly, it allows one to easily add in side information (meta data and etc.) without blowing up the computational speed.

5.1.2 Weighted Approximate-Rank Pairwise loss (WARP)

WARP is considering the pairwise strategy like BPR. In BPR, the SGD update with this difference as a weight. In WARP, SGD will be updated if the model predict wrong, for example, a wrong prediction means when the negative item has a higher score than the positive item. If the prediction will be correct, then the method keep redicting for both negative and positive items until it either get the prediction wrong or reach some cutoff value [Li 2017]. This method was proposed by Weston and et al. [Weston 2011] for the first time. They claimed that this process in addition of optimizing AUC like BPR, can optimize the precision@k too.

5.1.3 Alternating Least Squares (ALS)

The original loss function is actually a non-convex optimization problem but if we fix the set of variables X and treat them as constants, then the objective is a convex function of Y and vice versa. In another definition, in this method, we first estimate Y using X and estimate X by using Y. After enough number of iterations, we are aiming to reach a convergence point where either the matrices X and Y are no longer changing or the change is quite small.

5.1.4 Adaptive Moment Estimation (Adam)

The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications. In stochastic gradient descent, a single learning rate maintains for all weight updates and the learning rate does not change during training. However, Adam method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients.

In the following, I mentioned at the same time the state-of-the-art algorithms for matrix factorization and my consideration related to these methods to have more accurate result.

5.2 Methods

5.2.1 WRMF

Koren et.al. [Hu 2008] proposed a model looking into implicit feedback data to build a recommender for Netflix Grand prize[1]. This model improved a lot in other researches. The main algorithm later named in other researches as iALS, iMF or Weighted Regularized Matrix Factorization (WRMF) [Luo 2012]. WRMF is like the base of implicit matrix factorization. The novelty of this method has three aspects [Li 2017]:

1. The model treat the numerical value of implicit feedback to indicate positive and negative preference (binary feedback) related with extremely different confidence levels (weights).

2. Regarding the optimization strategy, a scalable optimization procedure, which scales linearly with the data size was proposed.

3. Regarding the generated recommendations by the factor model, a novel way to give explanations to the end user was proposed, which is rare among latent factor models.

The preference prediction function is given by $\hat{x}_{ui} = p_u.q_i^T$, where p_u and q_i are the user and item feature vector respectively. Observed binary preference p_{ui} is defined as

$$x_{ui} = \begin{cases} 0, & \text{if } r_{ui} = 0 \\ 1, & \text{if } r_{ui} > 0 \end{cases} \tag{5.4}$$

r_{ui} defines the observations for user behaviors (the numerical value of implicit feedback). As already mentioned, loss function of matrix factorization is the formula 5.3. WRMF is simply has a modification of this loss function:

$$L_{WRMF} = \sum_{u,i} c_{ui}(x_{ui} - p_u \cdot q_i^T)^2 + \lambda_p \sum_u ||p_u||^2 + \lambda_q \sum_i ||q_i||^2 \tag{5.5}$$

Instead of summing over elements of dataset (S) in formula 5.3, here it will sum over the entire matrix. We can consider ratings as binary preferences of the user; or we can consider the number of bookings of a hotel i for a user j as the ratings. In the WRMF loss function, the ratings matrix r_{ui} has been replaced with a preference matrix x_{ui}. We make the assumption that if a user has interacted with an item, then $x_{ui} = 1$. Otherwise, $x_{ui} = 0$.

c_{ui} is confidence matrix, and it describes how confident we are that user u does in fact have preference x_{ui} for item i. In the original paper [Hu 2008], one of the confidence formulas that they consider is linear in the number of interactions 5.6. If we take d_{ui} to be the number of times a user has booked a hotel, then

$$c_{ui} = 1 + \alpha d_{ui} \tag{5.6}$$

where α is some hyperparameter determined by cross validation. If we just consider binary "rates", so $d_{ui} \in \{0, 1\}$.

[1] https://www.netflixprize.com/

In Koren experiments, value $\alpha = 40$ produced the best results. It's necessary to notice that except the integrated c_{ui}, the calculation is over all the users and item pairs rather than only observed user and item pairs as in basic matrix factorization model, and because of that the loss function is quite dense, which contains $m \times n$ terms, where m is the number of users and n is the number of items. For usual datasets $m \times n$ can easily reach a few billions. This huge number of terms makes the direct optimization methods like SGD used for explicit feedback datasets not useful for implicit feedback datasets. Therefore, based on the observation that when either the user feature vectors or the item feature vectors are fixed, the loss function becomes quadratic so its global minimum can be readily computed, the authors proposed an improved ALS optimization process which integrates the confidence levels (c_{ui}) and overcomes the dense loss function.

In this model, we will just use the implicit data extracted from PNR transactions. However, for considering the context, we will do the post-filtering on the result. For example, if we want to see just the recommended hotels for one specific company in a specific month, we will apply three filters on our final result to have the expected result_ Company, City and Month. Since we filled all missing values, we are sure that our filtering will have the some result and it won't be empty.
Since post-filtering is not expensive from the company aspect, it can be considered as a good solution.

5.2.1.1 Implementation

There are a few number of implementations of this algorithm which easily can be find in github, among all of them the Python implementation[2] by Ben Frederickson, who wrote a pure Cython parallelized code, achieves 1.8 times faster than the multi-threaded C++ implementation provided by Quora QMF Library[3] and at least 60,000 times faster than implicit-mf[4]. For doing experiments in WRMF, Ben Fredericksons Python implementation was used.

5.2.2 LightFM

Kula [Kula 2015] present a hybrid matrix factorization model representing users and items as combinations of their content features latent factors. It is not a novel algorithm. It is a hybrid context-collaborative model which in its implementation has the capability to aggregate the metatdata into the related latent vectors. Then by considering metadata, this model can perform as a mixture of collaborative and content-based models for cold-start or sparse interaction data bases, and performs at least as well as a pure collaborative matrix factorization model where interaction data is available (warm-start, dense user-item matrix).
The features should be known in advance and represent user and item metadata. Users and items are fully described by their features. Exactly based on what is described in [Kula 2015], If we consider U be the set of users, I be the set of items, F^U be the set

[2] https://github.com/benfred/implicit
[3] https://github.com/quora/qmf
[4] https://github.com/MrChrisJohnson/implicit-mf

of user features, and F^I the set of item features, each user can be described by a set of features $f_u \subset F^U$ and each item can be described by a set of features $f_i \subset F^I$. The latent vector for each user feature f is described by $e_u^f \subset F^U$ and for each item feature f is e_I^f. Each feature is also described by a scalar bias term (b_u^f for user and b_I^f) for item features). The model takes binary feedbacks. The set of all user-item interaction pairs $(u, i) \in U \times I$ is the union of both positive interactions S^+ and negative interactions S^- .

The latent representation of user u is given by the sum of its features latent vectors:

$$p_u = \sum_{j \in f_u} e_j^U \tag{5.7}$$

The same holds for item i:

$$q_i = \sum_{j \in f_i} e_i^I \tag{5.8}$$

The bias term for user u is given by the sum of the features biases:

$$b_u = \sum_{j \in f_u} b_j^U \tag{5.9}$$

The same holds for item i:

$$b_i = \sum_{j \in f_i} b_j^I \tag{5.10}$$

The predicted preference for user u on item i is given by the dot product of user and item latent representations, modified by user and item feature biases:

$$r_{ui} = f(p_u.q_i + b_u + b_i) \tag{5.11}$$

Lots of functions can be suitable for f(.); For predicting explicit ratings, an identity function would work well. For implicit feedback, learning to rank algorithms can work well. For LightFM, BPR and WARP ranking losses which were explained in section 5.1 can be used.

The final model which we are trying to do it can be seen in figure 5.1. As it is obvious giving profiles information is optional in this model. If we use profile embedding , as I already explained, this information should be combined with the latent features from implicit data to have user and item representations. Finally, we can do the postfiltering for having specific recommendations for each context.

5.2.2.1 user/item profiles

Profiles should be known already for both users and items to be embedded in the representations of users and items. The only data which we have access is booking data which doesn't include any side information about user and items (hotel and company).
From the company side, even we did not have the name of the company, then one way is scrapping data from the web such as their Wikipedia pages but there are two limitations: 1) all of the companies don't have Wikipedia pages and 2) Information which are important for us like size and the field of company are not always mentioned in their pages.

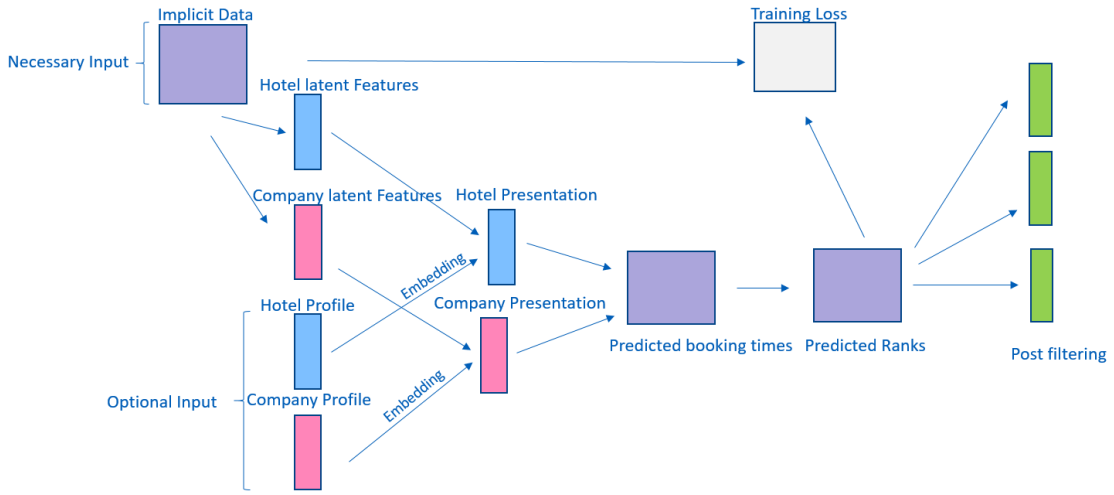


Figure 5.1: Final Model

From the Hotel side, we have extra information about their features but most of these features are like having swimming pool, the location based on the center and etc. are more related to the leisure travelers not business travelers. Then I just decided to create the profiles for both hotels and features from the implicit feedbacks which we have. The extracted features from PNR which were explained in chapter 3 are important features which can defines our item and hotel profiles. I just decided to use those features to create profiles for each hotel and for each user. I tried to create a profile which identify these informations:

1. Where are the main headquarters of each company
2. Size of the company
3. They have a specific discount rate or they use the discount negotiation rate of their partners

For the item 2 and 3, I tried to find this information based on these features:

- Based on their business, they need to have flights frequently or not
- What is the average expenses which each company considers for their business traveling
- They travel frequently in different months of years or mostly just during specific time

I have created a dictionary for each user which contain the following information with their weights, the default weight which I considered for them based on the expert opinion of the company was equal to one for all of them. The features for the company profile:

- 4-top most used rate codes per each company
- 4-top cities in number of traveling per each company

- Average price of booking hotels for specific company
- Number of business traveling times per each company
- All month of traveling
- The number of different city which a company travel to them

Year of traveling could be considered too but since our data is just for last seven months, then just considering months are enough.

The same holds for hotel profiles, this information should show why some specific hotels are important for just specific companies in one specific city. This information can be as following:

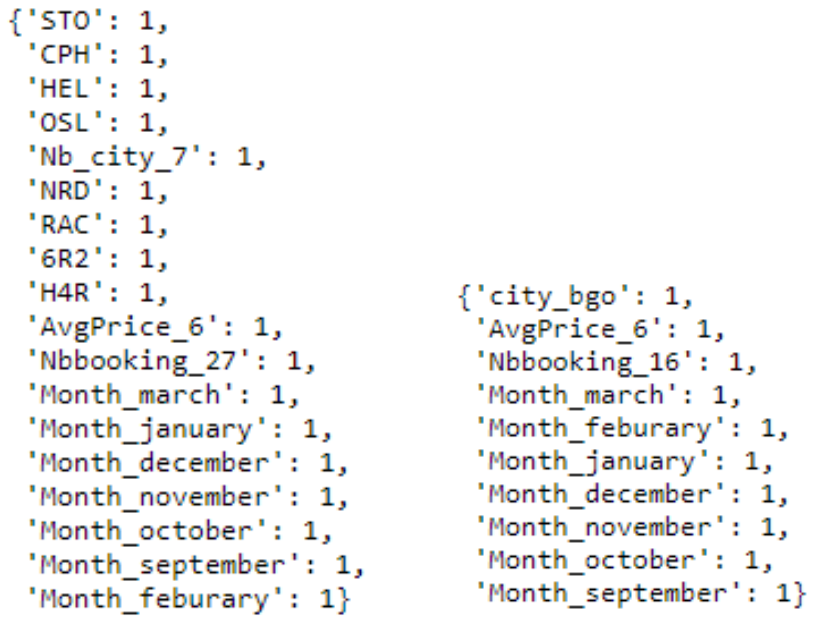
1. Average price of booking a specific hotel
2. All months which a hotel was booked in general (since some hotels don't have available rooms during specific time_e.g. high seasons)
3. The number of different company which booked the same hotel (It can show the distance of hotel to the offices in a city to this hotel as well as the preference)
4. The city where this hotel is located
5. The number of times which a hotel was booked in general

Finally, the same holds for features for the hotel profile:

- 4-top most used rate codes per each company
- 4-top cities in number of traveling per each company
- Average price of booking hotels for specific company
- Number of business traveling times per each company
- All months of traveling per each company

For example for a company like Nordea which we just extracted the name from business email from PNR transactions, we have this profile shown in figure 5.2, which I can interpret in this way that main headquarter of this company is in Scandinavian countries (I have STO, CPH, HEL, OSL city code which means Stockholm, Copenhagen, Helsinki and OSLO) and their business mission needs business traveling frequently (Nb_booking_27 which means category 27 - 27 is ceiling value of cube root of real number of booking-then the real value of number of booking is between 19683-21951) and they have enough money to spend for one night in expensive hotels (avg_price_6 means the money which this company pays in average for one night stay in a hotel is between 216-342 euros).

I had a look in Google to just confirm my profile, I can find that Nordea is a Nordic financial services group operating in northern Europe, the main headquarter is in Stockholm, Sweden. Also Nordea Bank AB commonly known as Nordea is part of Nordea -



(a) Company profile_NORDEA

(b) Hotel Profile_SCANDIC FLESLAND AIRPORT

Figure 5.2: Examples of created profiles for one company and one hotel

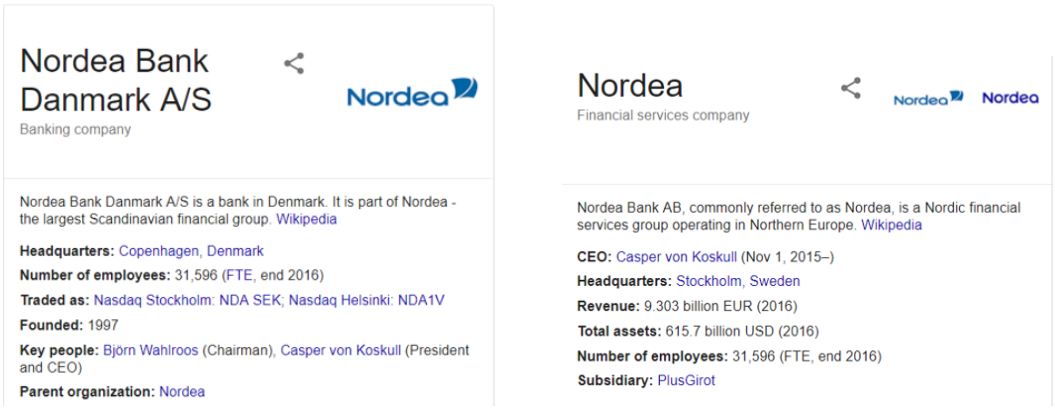
the largest Scandinavian financial group. The main headquarter of this bank is in Copenhagen, Denmark. The revenue of this company in 2016 was 9.303 billion Euro and the asset was 615.7 billion USD. Also they had 31,596 employee in 2016. Then I can confirm, with just using derived features from business policy features, I could have enough information to have a general overview about the company. This information can be seen in figure 7.9.

In addition, based on business policy, the 4-top most frequent rate codes for this company are (NRD,RAC,6R2 and H4R), which can identify the same companies which use the same negotiation rate codes. Also, They travel in all seven months of our historical data.

I vectorized these dictionaries for all of the companies to have the company profiles and after that I combined this company profiles with the latent company features which is extracted from implicit data to have a presentations for each company exactly like figure 5.1.

For the hotel is the same process and finally I have a profile like the figure 5.2 as the profile for each hotel. Then I vectorized these dictionaries to have a profile per each hotel and combine them with latent hotel features extracted from implicit data to have a presentations for each hotel exactly like figure 5.1.

One important issue is embedding metadata (side information) of the hotels and company as their profiles will give better results or just using the model by just considering implicit data will have a better result. For answering this question, we used this method for four different model as following:



(a) Nordea Bank (b) Nordea Financial Service

Figure 5.3: Information about Nordea from google

1. Model 1 : Considering implicit data as input without considering side information
2. Model 2 : Considering implicit data with just considering hotels profiles as input.
3. Model 3 : Considering implicit data with just considering companies profiles as input.
4. Model 4 : Implicit feedback with considering both side information as input (hotels and companies profiles).

Another important issue here is about defining important features in our profile (side information) . If I change the attributes in the created profile, Do the result will change? For example if I add four other items as following to my features in hotel profile and one more feature to my company profile, my final result will be biased to the profile which I chose or not.
New added features to hotel profiles:

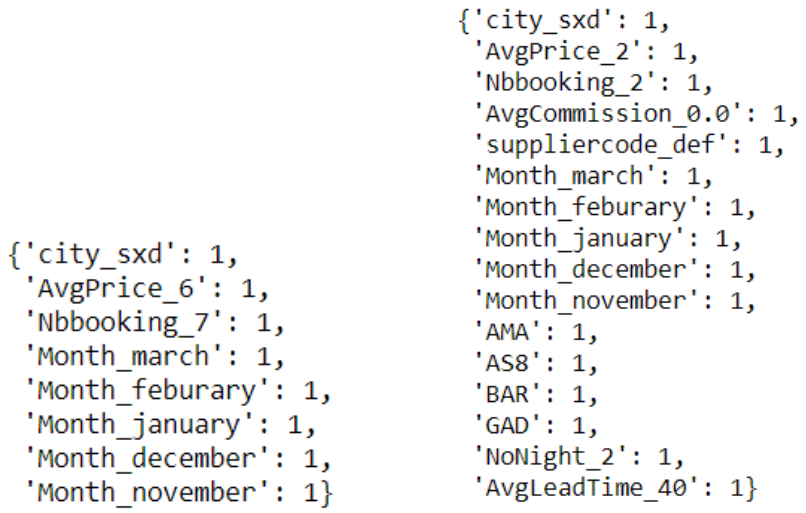
1. Supplier code for each hotel
2. Average commission rate between a hotel and travel agencies
3. average length of stay for each hotel
4. 4-top most used rate codes per each hotel
5. Average lead time to book a room in his hotel

New added features to company profiles:

1. average length of stay for each company

The new Profile for each hotel with new information will shown in figure 5.4 for one random hotel.

Based on the result in the experiment section, the results are biased to the known user/items profiles.



(a) First Hotel Profile (b) Second Hotel Profile

Figure 5.4: Two different hotel profiles for the same hotel_ *BEACHCOMBER FRENCH RIVIERA*

what I can do in this step is using deep neural networks leaning model on metadata embeddings to be sure we can choose the best profile based on the features which we considered them as the business policy. I explain this idea in section 5.2.3.

5.2.2.2 Implementation

As mentioned before, lightFM was introduced for the first time by Maciej Kula (who was working for Lyst - a fashion shopping website) in [Kula 2015]. LightFM is written in Cython and is paralellized via HOGWILD SGD (an update scheme to Parallelizing Stochastic Gradient Descent). He released a Python implementation of LightFM and made the source code for his paper and all the experiments available on Github.[5]. For embedding the features and contextualizing them, I used LightFM method. This implementation actually has more functions than what is described in the paper.

5.2.3 Collaborative Deep Learning (CDL)

Collaborative topic regression (CTR) is a recent proposed method taking this approach which tightly mixed the two components that learn from two different sources of information. It's a probobalistic graphical model that continuously integrates a topic model, latent Dirichlet allocation (LDA) and probabilistic matrix factorization (PMF). CTR is an appealing method but the latent representation learned is often not effective as they expected. Though, deep learning models recently used effectively for learning representations. Features in these models can be learned in a supervised or unsupervised method automatically which means effective feature representation is learned from other text contexts such as metadata. However, this methods has lower rank than shallow models such as

[5] https://github.com/lyst/lightfm/

CF in learning the similarity, especially when the dataset is so sparse, for solving this problem, collaborative deep learning (CDL) was developed by wang and et al. [Wang 2015]. It is a hierarchical Bayesian model, which performs deep representation learning for the content information and collaborative filtering for the utility matrix. CDL is using stacked denoising autoencoder (SDAE) for its feature learning component which is explained in section 5.2.3.1. CDL can simultaneously extract an effective deep feature representation from content (any kind of side information) and capture the similarity and implicit the relationship between items and users. This model can be shown in figure 5.5.

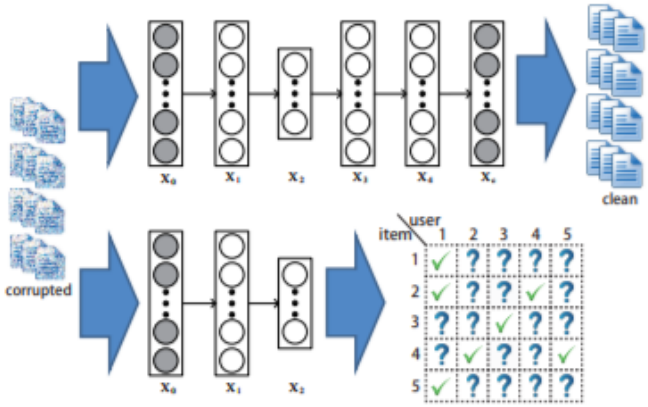


Figure 5.5: NN representation for CDL [Wang 2015]

As I mentioned before, CDL algorithm use PMF as base of matrix factorization which uses a denoising autoencoder to incorporate texts (side information) into the recommendations. Yang [Yang 2018] implemented a framework with name of Openrec which has three modules of Interactions, Extraction and Fusion components which which each of them apply different algorithms on several different source of information to prepare the best input for building a recommendation system. Details of these modules is available in original paper [Yang 2018].

The architecture of hypothetical music recommendation algorithm based on this framework can be seen in figure 5.6. The structure of this framework is shown in figure 5.7.

5.2.3.1 Stacked Denoising Autoencoders(SDAE)

SDAE is a feedforward neural network for learning representations of the data by learning to predict clean input itself in the output. Usually in a neural network, the hidden layer in the middle is constrained to be a bottleneck and the input layer is a corrupted version of the clean input. An SDAE solves the following problem:

$$min_{W_l,b_l}||X_c - X_L||_F^2 + \lambda \sum_l ||W_l||_F^2 \tag{5.12}$$

Where X_c is the clean input , X_0 is the corrupted input , X_l indicates each layer of network, W_l indicates weight matrix of each layer and b_l is the bias vector. If λ goes

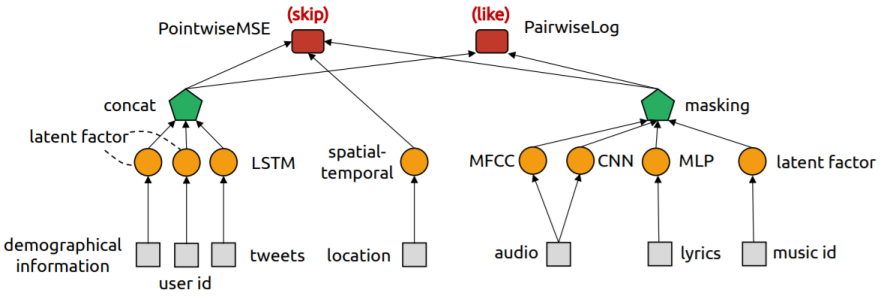


Figure 5.6: A hypothetical music recommendation algorithm [Yang 2018]

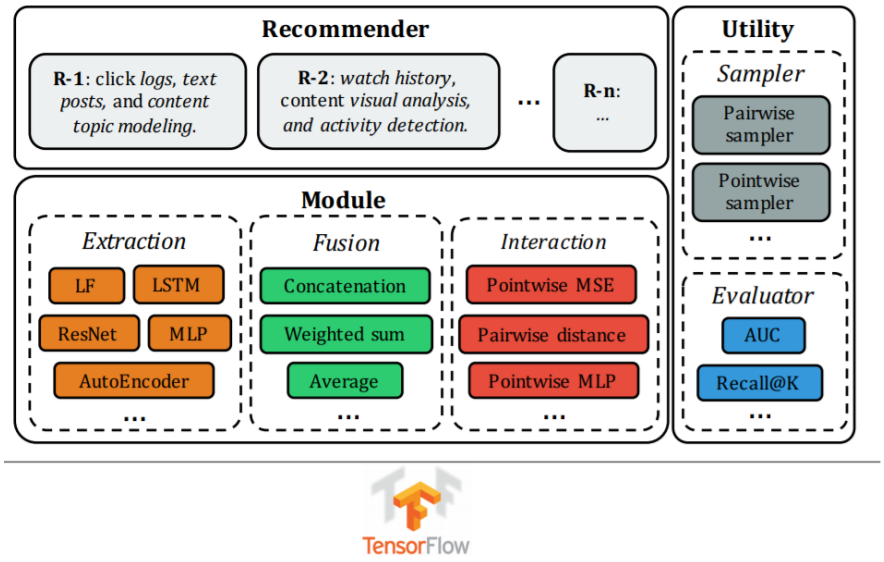


Figure 5.7: OpenRec framework structure [Yang 2018]

to infinity, the Gaussian distribution of each layer of network, will become a Dirac delta distribution centered at a function which is a sigmoid function. I'm not going to explain the math behind this model but you can see all of the details in these papers [Wang 2015, Vincent 2010].

5.2.3.2 Implementation

OpenRec is an open-source and modular Python framework library for neural network-inspired recommendation algorithms that supports lots of different recommender systems. The structure of this framework can be seen in 5.7. OpenRec is built to ease the process of extension and adaption state-of-the-art neural recommenders to heterogeneous recommendation scenarios, where different users, items, and contextual data sources need to be incorporated. OpenRec was introduced by Yang [Yang 2018]. The back-end of all the modules and algorithms are TensorFlow. He made the source code for his paper available

on Github [6]. I used this implementation to run the experiments for collaborative deep learning model. I used the modules : PMF and UserPMF.

[6] https://github.com/ylongqi/openrec

Chapter 6

Experiments

In this chapter, I will explain my experiments for the matrix factorization based models. The list of models and their features associated to them are mentioned in table 6.1.

For LightFM I considered eight different experiments with considering four different models, embedding user and item profiles separately or at the same time which I will explain them in following.

Also, for CDL algorithm, I just did two experiments, one just based on the historical information without any other auxiliary information and one with considering historical information and user profiles as input sources.

Finding hyperparameters is one of the most important task, before running the final model to tune the parameters in a way which model can give the best objective result. For finding the Hyperparameters, we should divide our dataset to train, evaluation and test dataset. we use the evaluation and train dataset to find the hyperparameters, then we set this hyperparameters in final model and we run the model for train and test dataset. We cannot randomly split the dataset since if for example I move 5 items from training to test for the users, then some of the users may not have any data left in the training set (Initially I just chose the users which has more than 5 booking) . Thus, the splitting dataset only looks for users who have at least 2*k (10 in this case) bookings before moving some of their data to the test set. This obviously biases our cross-validation towards users with more booking times. Also we just consider users in evaluation which they are in test set and we consider just one item per each one in evaluation test.

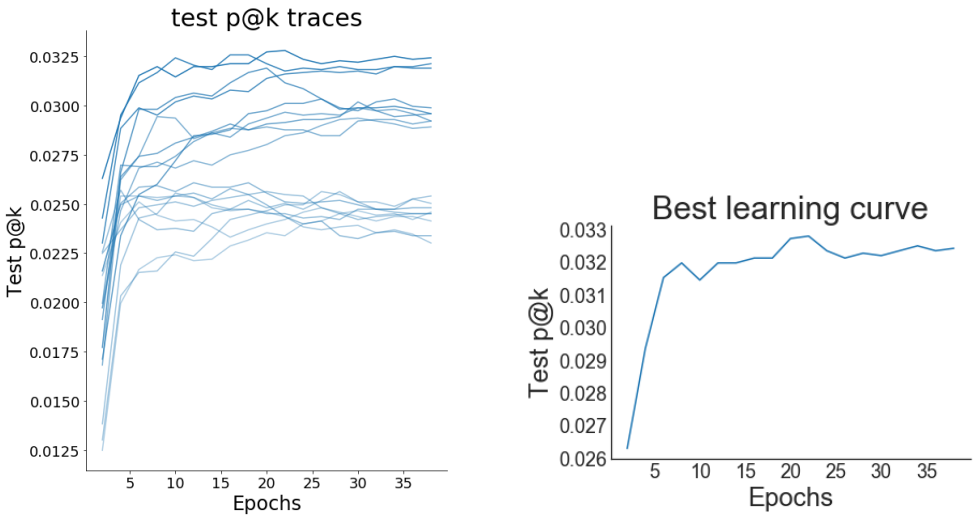
WRMF

I used the ALS as our optimization method, the parameters which need to be tuned are as following:

- num_factors: The number of latent factors, or degree of dimensionality in our model.
- regularization: Scale of regularization for both user and item factors.

CARS Method	MF Method	Optimization Method	Information	Strategy
Post-filtering	WRMF	ALS	Implicit Feedback	Pointwise
Post-filtering	LightFM	WARP BPR	Implicit Feedback Profiles	Pairwise
Post-filtering	PMF (CDL)	Adam	Implicit Feedback Profiles	Pointwise

Table 6.1: Different Experiment Models



(a) Test P@5 traces

(b) Best test P@5

Figure 6.1: Trace of evaluation P@5 for selecting Hyperparameters in ALS

- alpha: Our confidence scaling term.
- iterations: Number of iterations to run ALS optimization.

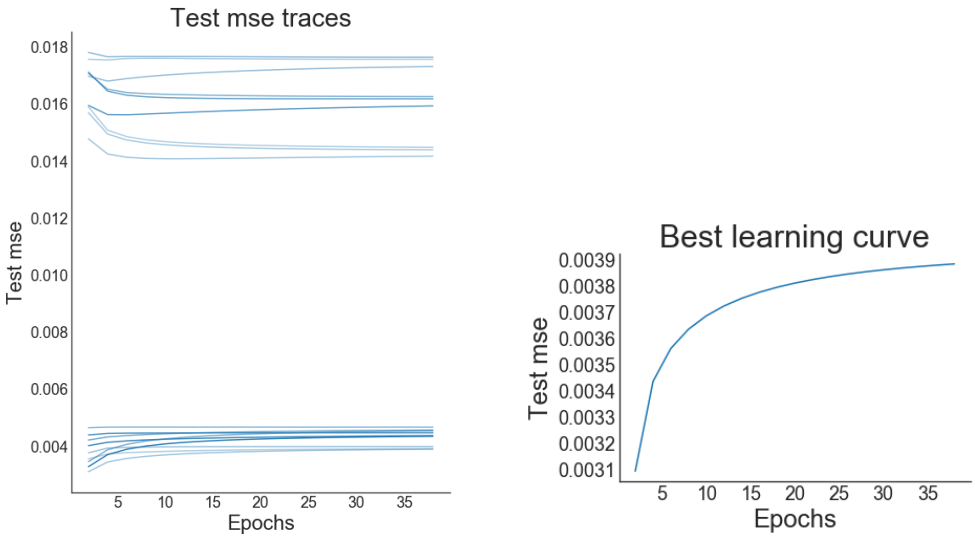
I used grid search technique to performing hyperparameter optimization.I kept track of mean squared error (MSE) and precision at k (p@k) for this algorithm, the result of all traces and best learning curve are shown in figures 6.1 and 6.2. The best results are:
(1) for p@k: 'num_factors': 50, 'regularization': 0.1, 'alpha': 10 and 'the best test p@k' : 0.0327979081060889 'Epoch': 22
(2)for MSE: 'num_factors': 50, 'regularization': 0.1, 'alpha': 10 and 'the best test MSE' :0.0030949327858624098 'Epoch': 2

The minimun mse and best p@k have the same value for these parameters: 'num_factors': 50, 'regularization': 0.1, 'alpha': 10

In this step, I was not sure that which metric should be selected as my optimizing object, then I finally found which best metric is AUC for implicit feedback based recommender system, then final considered hyperparameter for final models are: 'num_factors': 50, 'regularization': 0.01, 'alpha': 10 and 'the best test AUC' :0.84 'Epoch': 24

LightFM

I used BPR and WARP algorithm (Learning to Rank optimization algorithms). WARP does introduce two hyperparameters, though. One is the margin which determines how wrong the prediction must be to implement the SGD update. In the original paper [Weston 2011], the margin is 1 meaning you must guess puj>pui+1 to implement the SGD update. The other hyperparameter is the cutoff which determines how many times you're willing to draw negative samples trying to get a wrong prediction before you give up and move on to the next user. This sampling approach causes WARP to run quickly when it starts training because you easily get predictions wrong with an untrained model.



(a) Test MSE traces

(b) Best test MSE

Figure 6.2: Trace of evaluation MSE for selecting Hyperparameters in ALS

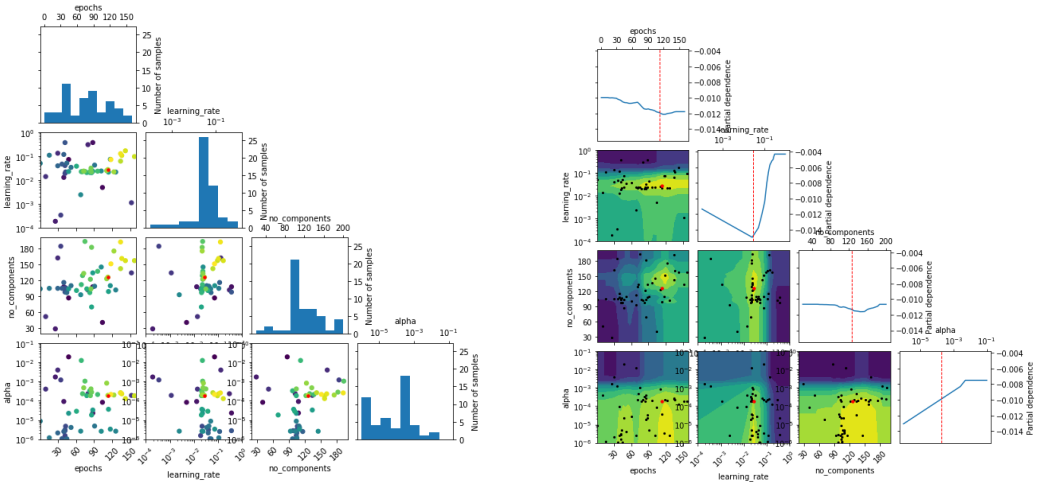
After training, though, WARP will run slower because it has to sample and predict many items until it gets a wrong prediction and can update. Sounds like a positive problem if it's hard to get your model to predict wrong, though. The parameters which need to be tuned for WARP are as following:

- no_components (no_factors) : The dimensionality of the feature latent embeddings.
- learning_rate : learning rate of the model
- alpha: L2 penalty on features (both item and user).

As I already mentioned, I run this method for four different models as following:

1. Model 1 : Implicit feedback without considering Profiles embeddings.
2. Model 2 : Implicit feedback with just considering hotels profiles embedding.
3. Model 3 : Implicit feedback with just considering companies profiles embedding.
4. Model 4 : Implicit feedback with considering both profiles embeddings(hotels and companies).

Figure 6.3 is visualizing hyperparameters which is chosen by bootstrapping sampling on all train dataset, I used extremely randomized trees method (known as Extra-Tree) proposed in [Geurts 2006] which is a variant of a random forest. Unlike a random forest, at each step the entire sample is used and decision boundaries are picked at random, rather than the best one. Actually the method drops the idea of using bootstrap copies of the learning sample, and instead of trying to find an optimal cut-point for each one of the K randomly chosen features at each node, it selects a cut-point at random. From a statistical point of view, dropping the bootstrapping idea leads to an advantage in terms of bias, whereas the cut-point randomization has often an excellent variance reduction effect. This



(a) Bootstrapping sampling (b) Partial Dependency

Figure 6.3: Hyperparameters selection by bootstrapping sampling for WARP algorithm for Model 1

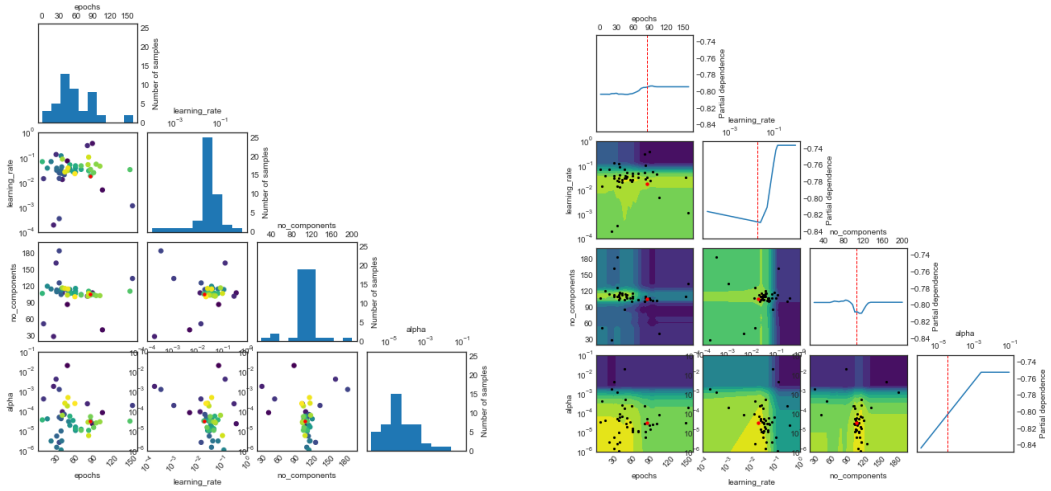
method has yielded state-of-the-art results in several high-dimensional complex problems. First of all, I was thinking which p@k is a good objective function like what I did in WRMF algorithm but since it's a context-aware RS, the top 5 hotels are for all of the cities, not considering per each city and our preference matrix is binary (implicit feedback based RS); then considering p@k it's not a really good objective function. Also, my preferences matrix in the model is a binary model which is necessary for me to keep the balance between the two class of positive and negative items (booked or not booked). Then based on the literature review, the AUC score seems a good objective function. Then in figure 6.4, I did the process once more with considering AUC (area under curve) as the objective.

The color of the area or the dots shows the order of choosing the samples, blue dark is the earliest point and the yellow part shows the later sampling parts. Also, The red one shows the best combination part of two dimension to have the best objective value.

The result can be seen in table 6.2.
For the model 2, I used the best hyperparameter for the model 1 as the initial point to search for better hyperparameters. This time I run the model just for 20 times with 160 epochs per each one. The output of all search area can be seen in figure 6.5

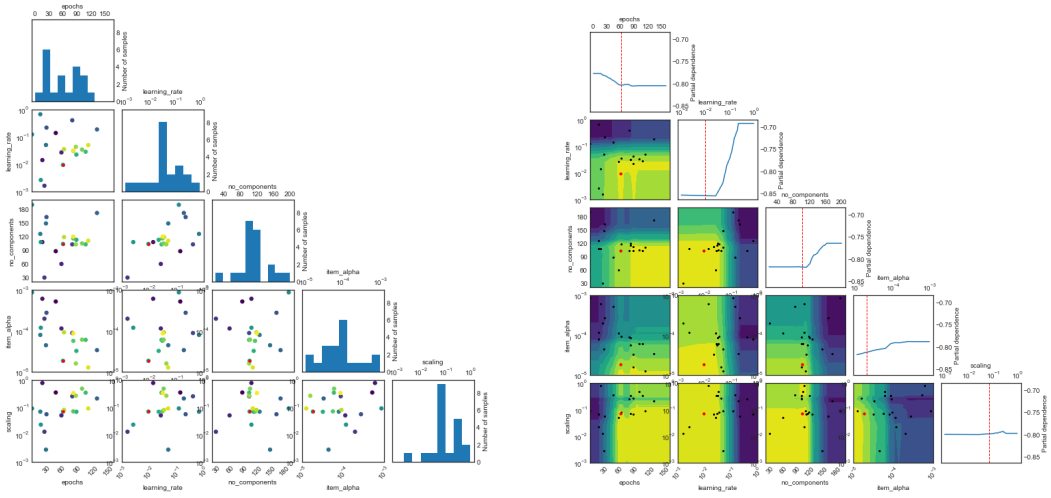
For the model 3, again I used the best hyperparameter of the model 1 as the initial point to search for better result. I run the model again just for 20 times instead of 50 times but as you can see in the figure 6.8, the changes after call 20th are so smooth, then we are not expecting a really big changes.

And finally, for the model 4, I again used the best hyperparameter of the model 1 as the initial point to search.Here again I just had 20 calls. The first reason is because learning by embedding decreased the speed of learning and because the initial model is a quite good recommendation system, it takes lots of time to have a wrong guess and in last iteration, each iteration takes lots of time. Also, in figure 6.8 you can see after call 5 or 6, we will have the minimum objective value.



(a) Bootstrapping sampling (b) Partial Dependency

Figure 6.4: Hyperparameters selection by bootstrapping sampling for WARP algorithm for Model 1_considering AUC as the objective function

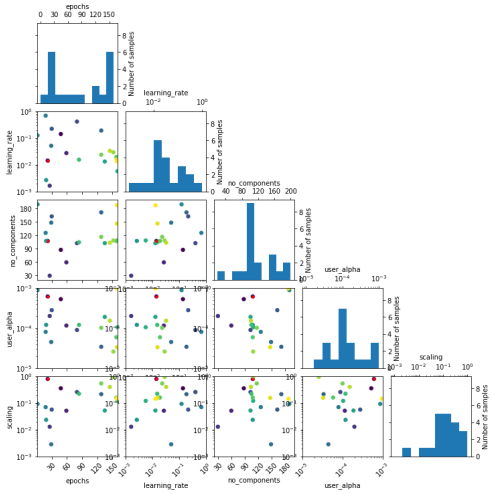


(a) Bootstrapping sampling (b) Partial Dependency

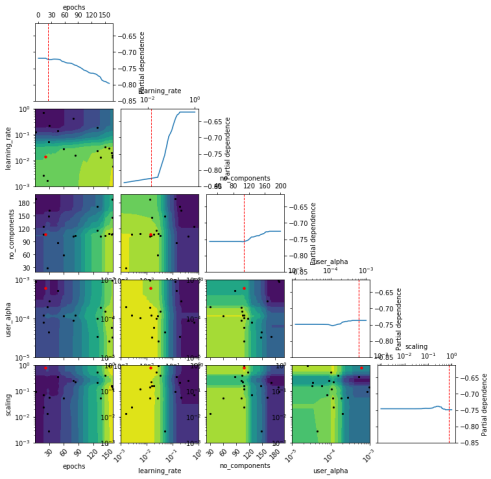
Figure 6.5: Hyperparameters selection by bootstrapping sampling for WARP algorithm for Model 2_considering AUC as the objective function

Model	Obj. value	epochs	learning_rate	no_factors	alpha	scaling
Model 1	0.91365	85	0.01845	106	3.184e-05	-
Model 2	0.92175	62	0.00950	103	1.856e-05	0.0697
Model 3	0.91330	22	0.01422	107	6.075e-04	0.77801
Model 4	0.92963	22	0.01422	107	6.075e-04	0.77801

Table 6.2: Hayperparameters for different LightFM models for WARP algorithm for evaluation dataset

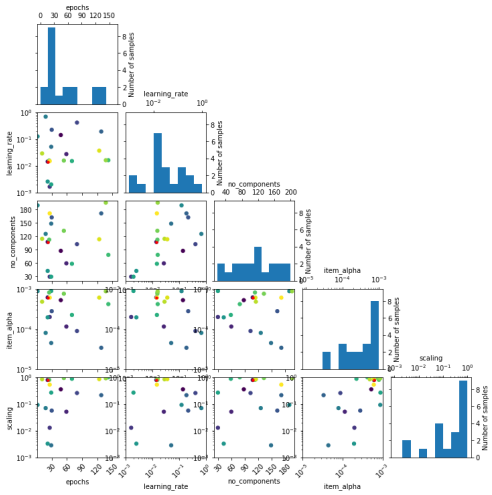


(a) Bootstrapping sampling

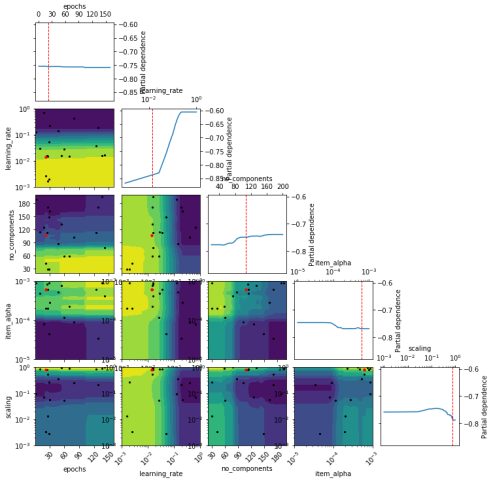


(b) Partial Dependency

Figure 6.6: Hyperparameters selection by bootstrapping sampling for WARP algorithm for Model 3_considering AUC as the objective function



(a) Bootstrapping sampling



(b) Partial Dependency

Figure 6.7: Hyperparameters selection by bootstrapping sampling for WARP algorithm for Model 3_considering AUC as the objective function

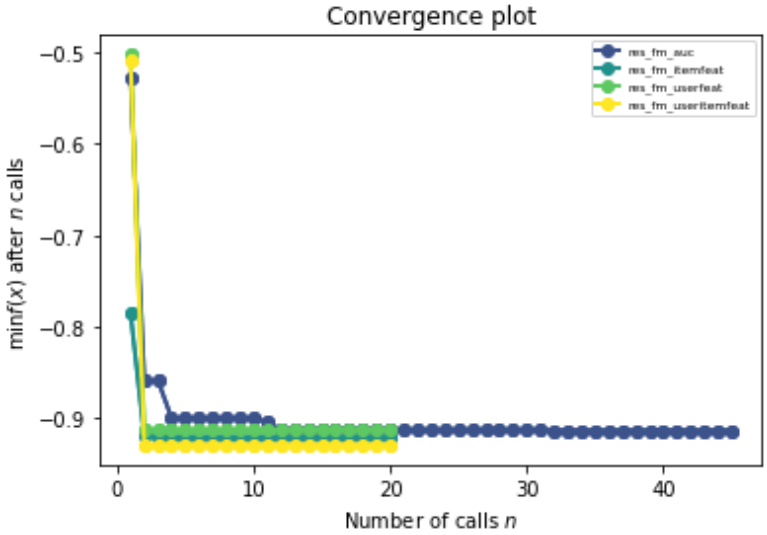


Figure 6.8: Comparison of AUC based on forest minimizing method for the 4 different model_WARP algorithm

Model	Obj. value	epochs	learning_rate	no_factors	User_alpha	Item_alpha
Model 1	0.87634	12	0.0086	136	3.32e-09	4.471e-09
Model 2	0.8946	24	0.01788	29	4.572e-09	3.742e-09
Model 3	0.8897	44	0.01422	34	3.21e-09	4.99e-09
Model 4	0.89908	23	0.00294	28	1.511e-09	2.907e-08

Table 6.3: Hayperparameters for different LightFM models for BPR algorithm for evaluation dataset

For BPR algorithm, I used the grid search CV for finding the best parameters, the object function is again AUC score. The Parameters and their value for each model is shown in table 6.3 .

Just from hyperparameters, it's obvious that WARP has a better result than BPR algorithm based on evaluation dataset.

Also for both algorithms, the model with embedding both profiles have better results. Though, this improvement is not considerable significantly but it was something which we were expected since the derived profiles from booking data already were considered in creating latent factors for each item and user but with embedding derived profiles, we give more weight to the features which are more important to us, then this smooth increase is expected. Though, if we can make the profiles from external sources, it's expected to have more effect on increasing AUC score.
For having a better understanding about the results, it's better to see the evaluation section.

Collaborative Deep Learning

For this method I just used the hyperparameters from the original algorithms [Yang 2018, Wang 2015]. Also, I tried to consider the best hyperparameter considered for previous MF methods for this method. The chosen optimization algorithm is Adam and the chosen matrix factorization algorithm is PMF which use pointwise strategy for sampling. The batch size for sampling is considered 1000, the number of latent factors is 107 and the regularization term (l2_reg) is equal to 0.01 for PMF method and for USERPMF we will have just l2_reg_mlp parameter which is the learning rate of the neural network on content embeddings. For SDAE algorithm from extraction module, the feature size of each encoding layer's output should be defined, dims=[512, 258, 128] creates an three-layer encoder with output shape [*, 512], [*, 256], and [*, 128], and a two-layer decoder with output shape [*, 256] and [*, 512]. Number of factors for matrix factorization should be equal as dimension of first layer encoder.

Configuration

All of the experiments was running locally by using CPU with 16 GB RAM and 7 core of 2.9 GHZ. All of the implementation is done by python 3.6.

Chapter 7

Evaluation

Before we start to evaluate and compare the different models in Chapter 6, we need to know what we want out of a recommendation system. It might seem obvious what distinguishes a good recommendation system from a bad one. A good recommendation system should recommend items for which user finds some interest to use or to buy. The more interesting the item is to the user, the better it is. User satisfaction is extremely important to the recommender system provider.

For our problem, there is however, no doubt that the most important objective is to recommend hotels to the business travellers which fit on the business policy of their company. Since we do not have access to the list of these hotels per each company to distinguish relevant from irrelevant hotels, we should choose a baseline to can evaluate the performance of our recommender systems. Since if a hotel was chosen in the past frequently can give us this information which this hotel is fit with the business policy and more likely to be liked by the user more than a hotel with less number of bookings; I chose popularity ranking as the baseline for considering if an item is relevant or irrelevant. Then if I chose top k recommended hotels for a user, if they are in k top most booked hotels for the same user (most popular hotels), it's considered as relevant but if it's not, it's irrelevant. This way of evaluation for our problem may seem not to be a good way because of two reasons:

1. The utility matrix was so sparse, then based on the learning method to filling the missing values, there is a high possibility which the hotels which can fit inside the business policy have a higher rank rather than those they were already booked by the user.

2. The final evaluation method here, does not consider the top k hotels per each city (the obligatory considered context here). It's just considered the rank of whole hotels in all cities per each user. Evaluation per each context is not easy since I have a high cardinality per each context and I should calculate the relevant and irrelevant items per each city per each user which it's complex and time consuming.

Based on these limitation, best evaluation methods for the final chosen recommendation system should be online or user study experiments. However, for finding the best models among all of the experiments which we have, we can trust on the offline evaluation methods with considering the relevant and irrelevant items in the way which I have explained.

Since our problem in this research is a top K recommendation task which Amadeus wants to give just top 5 recommended hotels to each business travellers and based on the introduced evaluation metrics in section 2.10, I chose these metrics to compare the models

which we experimented them in chapter 6: MAP@5, Mean average recall@5, MRR@5, Mean AUC score. Result is shown in figure 7.1, 7.2, 7.3 and 7.4. In the following I'm just using these terms instead of each time mentioning they are average value of metrics: MAP@5, recall@5, MRR@5 and AUC. As a benchmark, I also calculate what these metric value would have been if we had simply recommended the most popular items to each users. Popularity tends to be hard to beat in most recommender system problems, so it makes a good comparison.

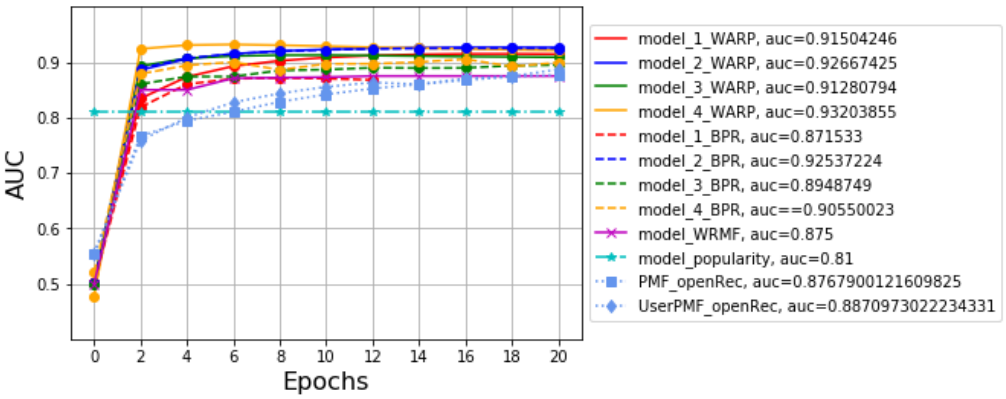


Figure 7.1: AUC score comparison

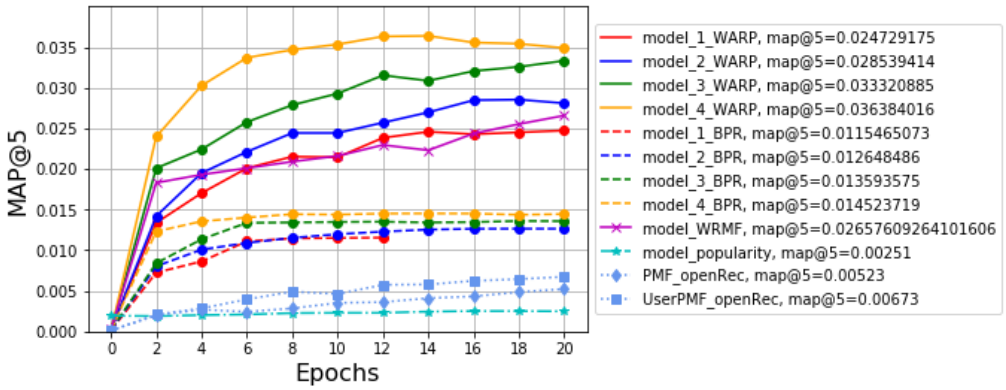


Figure 7.2: MAP@5 value Comparison

It is obvious that our recommender system have trained and proven it beats the benchmark of popularity for AUC score. AUC score is already high for popularity ranking method and it's not easy to beat. An AUC score of 0.93 means the recommendation system model is recommending hotels which the user in fact had booked in the test set far more frequently than hotels the user never ended up booking.
Among all of the models, metrics for MAP@5, AUC, R@5 and MRR@5 for model_4_WARP have higher values than others. In Particular, WARP has a better performance than BPR algorithm but in both of them, embedding both profiles in learning process has a better performance. This fact is the same for using neural network method for having better profiles for users. However, the performance of this model is

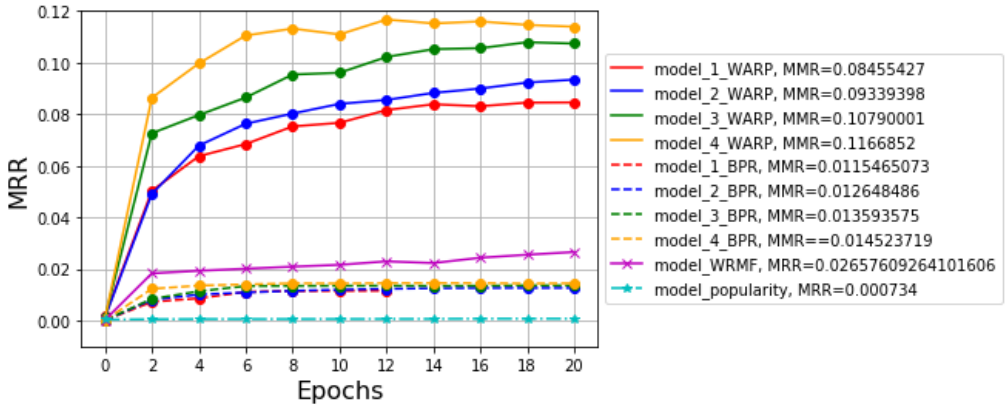


Figure 7.3: MRR@5 value comparison

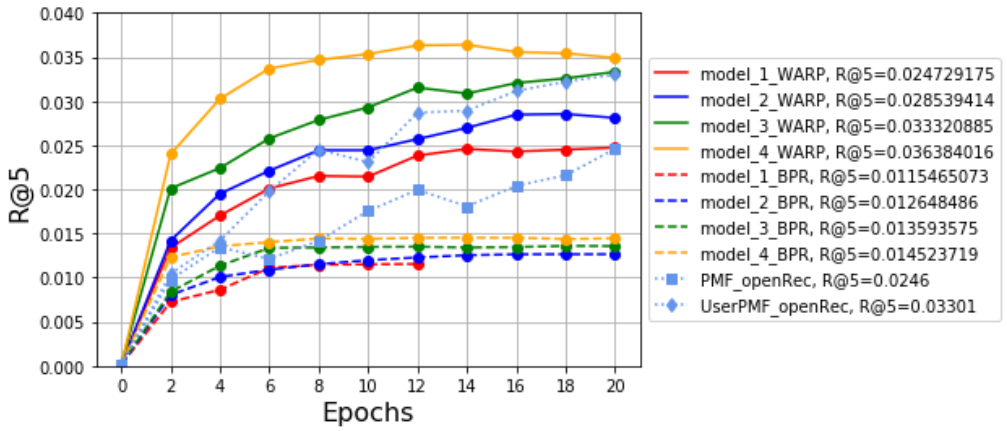


Figure 7.4: R@5 value comparison

much worse than just using embedding without doing selection on them. since PMF and UserPMF have different architecture in their models, I'm not comparing them directly with our final model. However, by considering them as two new experiments, they don't have better result than model_4_WARP.

Experiments by using neural network on top of the matrix factorization to choose better profiles did not have a good result here which can be caused by these reasons: 1) Choosing the hypereparameters just based on the original papers not based on the data 2) Initial profiles as input to the neural network model are not that much complicated and do not need a complicated model to clean them.

On the other hand, Amadeus is searching just for a long list of hotels which can keep them in memory without cashing them everytime, I considered top 100 hotels per each company. Also, I compared the same evaluation metrics for N=100: MAP@100, MRR@100,recall@100 and AUC.

The result is shown in figures 7.5,7.6,7.7 and 7.8. As we expected, since AUC is not dependent to number of K, the AUC score with increasing K is the same. But with increasing K, since lots of our user's didn't already booked 100 hotels, then it was expected

that the MAP@100 will decrease rather than MAP@5. Furthermore, since we recalling more relevant hotels, the recall@100 value increases rather than recall@5 . For the MRR, since it's just considering the rank of hotels per each user which already booked, and for lots of new recommended hotel, they were not already had a rank, then they were not considered in this evaluation, then, we do not expect a really big changes.

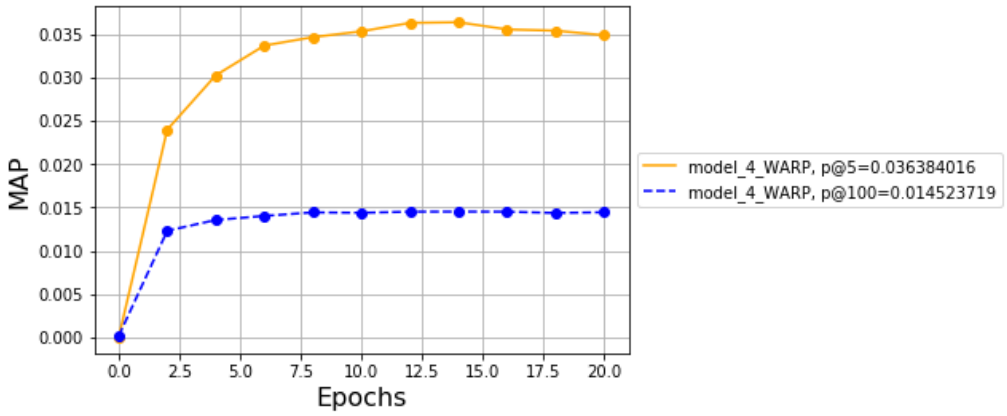


Figure 7.5: MAP comparison by increasing K

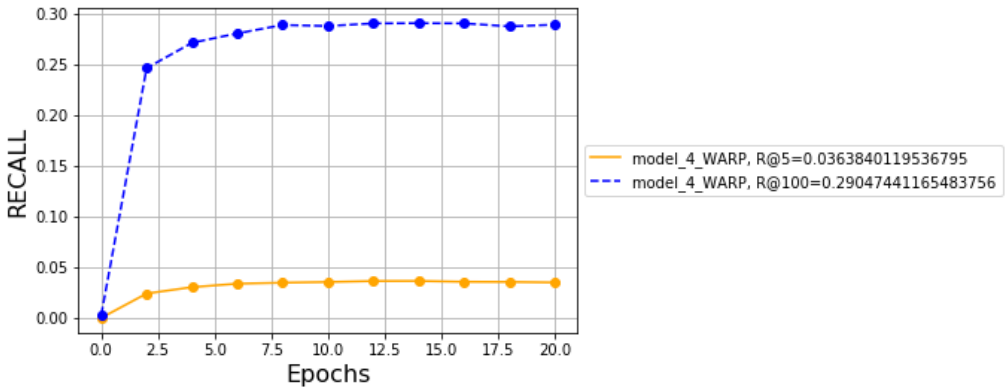


Figure 7.6: Recall comparison by increasing K

Final business policies are still unknown, they are considered inside the user and hotel representation. In the following the similar companies based on similar business policies from our method is shown in figure 7.9. For some of this companies, since the similar companies are part of a company (like IT Amadeus, Benelux Amadeus, Pixel, Navitaire and different branches of Amadeus (Dutch, UK and etc) understanding that they have the same business policies is easier. This similarity is based on company presentation (combining the latent factors with company profile vectors) derived from model_4_WARP.

Now which we know model_4_WARP has the best performance, we can run post-filtering on the final result. The obligatory context which result should be filtered based on that is city.

For evaluating the context city, we chose 8 different cities randomly but by considering which they were be distributed in most traveled city, middle and less traveled one. Also,

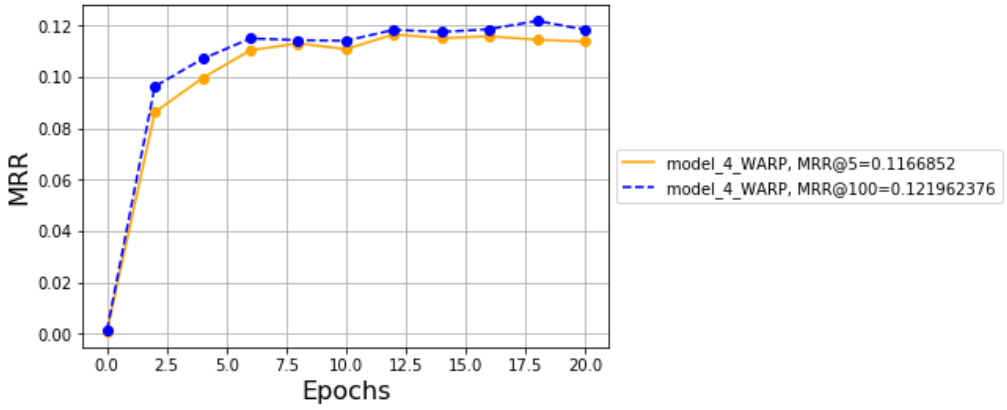


Figure 7.7: MRR comparison by increasing K

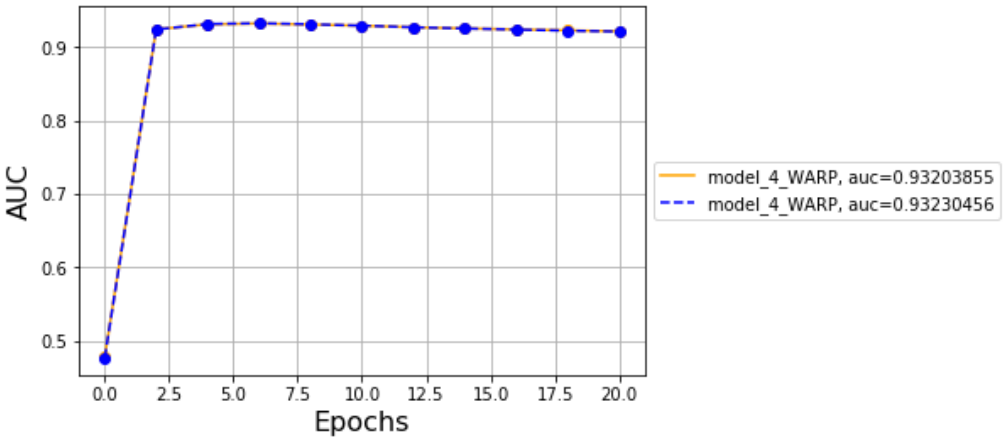
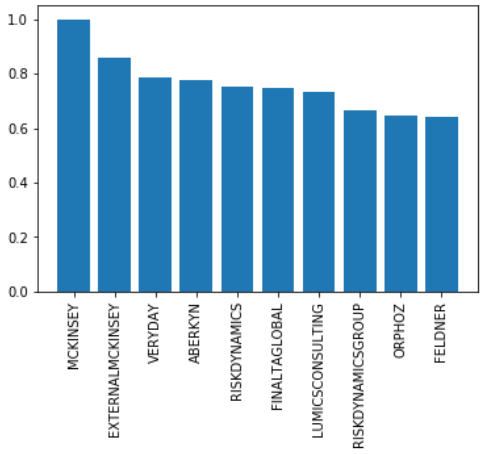


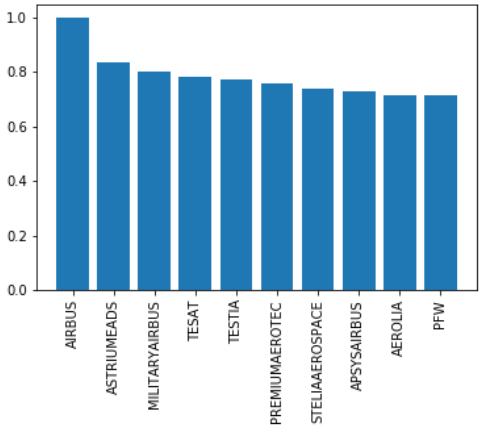
Figure 7.8: AUC comparison by increasing K

we chose 8 different companies as well. The number of transaction per each company and the number of travel per each city can be seen in figure . Then we will compare the result of our recommended hotel per each pair of company, city with popular ranking result by considering context city. Result can be seen in table 7.2 . we just compare the first 5 recommended hotels. As you can see in the table, based on the popular ranking, the recommended hotel are less than 5 hotels and in several cases, the recommended hotel based on popular ranking is null, because in the historical booking dataset, there was not any booking from these companies in this hotels. Which it shows if we add new items for each company, in our new model we have recommendations. It shows our model will solve the cold start problem for new items per each user. Value of p@5 and R@5 is shown for each pair of (company, city).

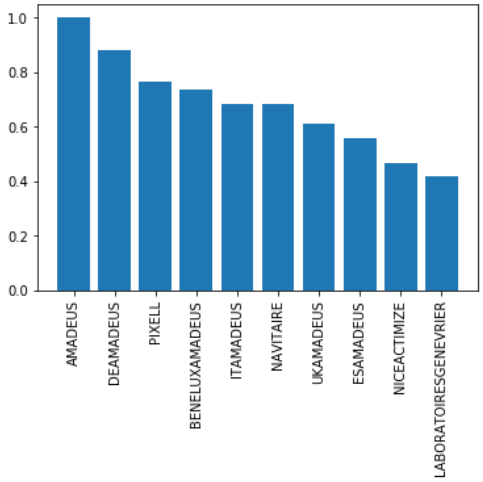
This evaluation is not comprehensive but can give an idea about how the final model based on the obligatory context 'city' can perform. Also, for considering other contexts, we can combine the final result with OLAP architecture defined in chapter 4.



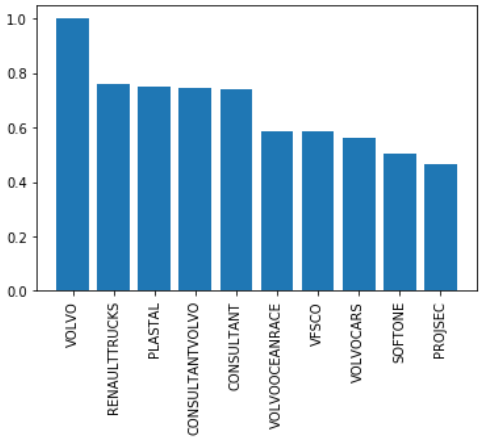
(a) Mckinsey



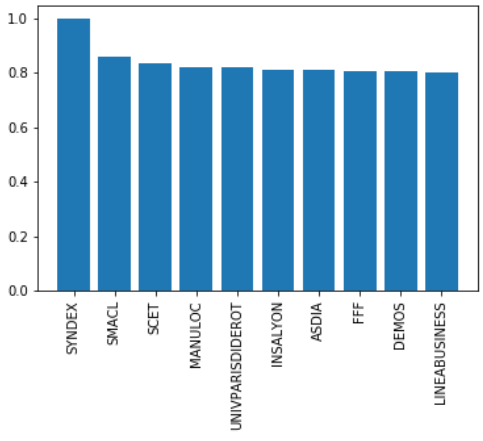
(b) Airbus



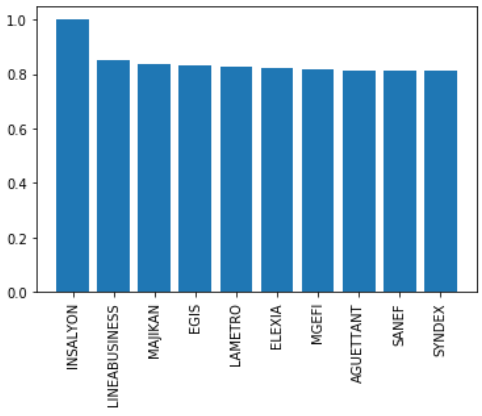
(c) Amadeus



(d) Volvo



(e) Syndex



(f) InsaLyon

Figure 7.9: Similar companies based on similar business policies

Table 7.1: Number of travels per each chosen city and company

Company	**Number of travels**	**City**	**Number of travels**
Mckinsey	62312	Paris (PAR)	112251
Airbus	34098	London (LON)	110802
BCG	22288	Stockholm (STO)	88557
Volvo	3948	Oulu (OUL)	4459
Renault	11445	Doha (DOH)	2920
DHL	4676	Bahrain (BAH)	991
Cisco	361	Montevideo (MVD)	510
Adidas	42	Marbella (QRL)	463

Table 7.2: Case study to compare the result from final model (LightFm_both profile embedding) by popular ranking

Company	City	Interactions	Final model	Popularity model	P@5	R@5
Mckinsey	Paris	1900	**166**, 1022, 1442, 3204, 3963	164,**166**,517,823,863	0.2	0.25
Airbus	Paris	1063	3283, 3271, 10810, 2061, 1175	2237,2735,2937,863,4148	0	0
BCG	Paris	487	16239, **4916**, 1022, 164, 16943	166,1676,164,**4916**,2982	0.2	0.2
Volvo	Paris	32	**9322**, 25443, 547, 16654, 30914	4148,**9322**	0.2	0.5
Renault	Paris	171	22745, 4383, **2735**, 9424, 2754	**2735**, 4922,4070,6355,4383	0.2	0.2
DHL	Paris	38	22446, 148, **5633**, 21022, 158	2004,**5633**, 3372	0.2	0.33
Cisco	Paris	0	3178, 164, 20905, 4441, 1442	-	0	0
Adidas	Paris	0	14145, 3600, 15256, 17730, 1258	-	0	0
Mckinsey	London	3994	11650, 452, **113**, **75**, 13	3372,**75**,**113**,211,353	0.4	0.4
Airbus	London	492	9606, 17193, 31592, 9009, 2629	2393,1550,1973,3602,3943	0	0
BCG	London	1226	**13**, 20772, 1893, 5592, 21517	387,922,311,882,**13**	0.2	0.2
Volvo	London	6	1876, 15054, 873, 15497, 3430	1367, 5045,444,8679,1417	0	0
Renault	London	8	3042, **1972**, **6783**, 1007, 56	**1972**,4524,4168,8362,**6783**	0.4	0.4
DHL	London	44	10607, 434, 1506, 1876, 1973	1664,2530,461,19200,14940	0	0
Cisco	London	1	928, 3319, 922, 1973, 387	1071	0	0
Adidas	London	0	452, 3095, 22998, 6419, 31500	-	0	0
Mckinsey	Stockholm	1525	**296**, 569, 20736, **30**, **193**	**30**,**296**,77,**193**,1469	0.6	0.6
Airbus	Stockholm	80	**2022**, 193, 240, 1172, 3270	77,3217,**2022**,9073,240	0.2	0.2
Mckinsey	Oulu	0	4274, 2229, 6773, 639, 19272	-	0	0
Airbus	Oulu	0	1633, 19272, 4274, 17911, 639	-	0	0
BCG	Oulu	0	4274, 2229, 6773, 639, 19272	-	0	0
Volvo	Oulu	3	6773, 639, 17911, **2229**, 19272	**2229**	0.2	1
Renault	Oulu	0	639, 19272, 2229, 6773, 4274	-	0	0
DHL	Oulu	0	1633, 19318, 19272, 4274, 639	-	0	0
Cisco	Oulu	0	1633, 4274, 6773, 30964, 639	-	0	0
Adidas	Oulu	0	6773, 30964, 4274, 17911, 19318	-	0	0
Mckinsey	Doha	215	**1862**, **2067**, 15696, **3960**, 982	**1862**,**2067**,**3960**,13712,982	0.6	0.6
Airbus	Doha	290	**2097**, 10989, 31510, **5983**, 2067	982,**2097**,**5983**,4319,5252	0.4	0.4
BCG	Doha	40	**1862**, **2067**, 982, **2097**, **12187**	**2067**,**1862**,**2097**,**12187**	0.8	1
Volvo	Doha	0	6111, 1108, 2097, 5252, 24934	-	0	0
Renault	Doha	0	15669, 31510, 15692, 12187, 24934	-	0	0
DHL	Doha	0	12187, 2097, 1108, 6111, 5252	-	0	0
Cisco	Doha	0	6111, 2067, 5252, 2097, 4060	-	0	0
Adidas	Doha	0	6111, 2067, 29241, 29082, 5983, 15696	-	0	0
Mckinsey	Bahrain	4	**15303**, **16699**, **7436**, 9076, 16727	**7436**,**15303**,**16699**	0.6	1
Airbus	Bahrain	15	**15303**, **12288**, 16727, 7436, 6196	**12288**,**15303**	0.4	1
BCG	Bahrain	1	15303, **2604**, 9076, 7436, 14038	**2604**	0.2	1
Volvo	Bahrain	0	15303, 6695, 18654, 15474, 14038	-	0	0
Renault	Bahrain	0	18654, 15474, 2604, 7436, 12288	-	0	0
DHL	Bahrain	4	14038, **6695**, **12288**, 15303, 28539	**12288**,16699,**6695**	0.4	0.7
Cisco	Bahrain	0	15303, 2604, 6695, 7436, 16699	-	0	0
Adidas	Bahrain	0	29588, 28539, 15303, 15474, 9076	-	0	0
Mckinsey	Montevideo	1	**9119**, 12732, 14573, 20693, 6323	**9119**	0.2	1
Airbus	Montevideo	9	**6323**, **9119**, 14573, 27785, 20693	**6323**,**9119**	0.4	1
BCG	Montevideo	0	9119, 14573, 21915, 20693, 30198	-	0	0
Volvo	Montevideo	0	20693, 14573, 12732, 29728, 21915	-	0	0
Renault	Montevideo	0	1915, 23266, 6323, 12732, 14573	-	0	0
DHL	Montevideo	0	20693, 27785, 12732, 23266, 21915	-	0	0
Cisco	Montevideo	0	6323, 14573, 30198, 14989, 27785	-	0	0
Adidas	Montevideo	0	6323, 14573, 21915, 29728, 12732	-	0	0
Mckinsey	Marbella	0	25518, 27960, 28878, 30967, 9241	-	0	0
Airbus	Marbella	0	11296, 9241, 27960, 30967, 14287	-	0	0
BCG	Marbella	0	27960, 25518, 9241, 28878, 30967	-	0	0
Volvo	Marbella	0	9241, 27960, 30967, 14287, 25518	-	0	0
Renault	Marbella	0	28878, 9241, 30967, 25518, 11296	-	0	0
DHL	Marbella	0	9241, 30967, 28878, 27960, 25518	-	0	0
Cisco	Marbella	0	9241, 28878, 27960, 14287, 11296	-	0	0
Adidas	Marbella	0	9241, 25518, 27960, 28878, 11296	-	0	0

Chapter 8

Conclusion

A recommendation systems technique should be chosen carefully based on the problem. There is not a global solution or method for recommendation systems. For choosing the appropriate technique, a few issue should be defined, first which kind of data we have access and then what are the most important components for us to be considered in our problem (popularity, novelty and etc.). This component defines an important role in evaluating the recommendation performance. The sparsity of utility matrix and even the amount of first data without any filtering have a huge effect on choosing the proper method.

In this thesis, we have extended the 2D matrix factorization method for implicit feedback datasets to consider the context and content at the same time. This way we can consider context in a recommender system to have more accurate results. We explored the recommendation systems methods and their advantages and disadvantages and why matrix factorization is a good method for our solution to build a recommendation system. Furthermore, based on context dimensions, we thought to integrate recommendation system with OLAP method as another different solution. Since post-filtering is not expensive from industry side, and we can consider any added contexts in future, among post-filtering, pre-filtering and contextual filtering, I chose post-filtering method to corporate context-awareness. While most recommenders use explicit feedback, we did not access this kind of data and we focused on implicit feedback which was extracted from historical booking information. Also, the name of each user (company name) was not defined per each transaction, then we extracted the name of the companies from their business email addresses. Finally we listed important evaluation methodologies and explained which one of these metrics can be used for measuring implicit feedback based recommender systems. We compared the result of some different implemented methods with four different evaluation metrics and the best one was the one with using WARP algorithm to improve the loss function and embedded extra profiles was learned at the same time through this algorithm. Since we did not have access to external information to create profiles for hotels and companies, for creating profiles, we derived some new features from booking information and we considered a set of this new derived features as the profiles of users and items. The drawback of this profiles is that we can build different profiles and there is not a global profile for hotels and profiles, then results can change by changing the profiles. This change are not that much considerable, since all of the features of these profiles derived from historical data which already were considered in building the latent factors but still these new profiles give more weights to the item mentioned in the profiles. The best model which has better result based on all chosen evaluation metric is the lightFM model by considering both Hotels and Companies profiles (Model4_WARP). I had this question in my mind: Can we do a selection on finding the best combination of features from profiles as our input to our model? We used a neural network model for finding the best combination of features as the Input of our

model. Since I just gave the same profiles to neural network model, we can see that still we have improvement in result compared with just using matrix factorization but not compared to final result of final model. Since matrix factorization in this model uses the bpr as optimization algorithm, we compared the result with the the same optimization model but still result with neural network model is much worse which can be because of choosing the hyperparameters not based on the data, but based on the original papers. Also, it can be because of simplicity of the profiles as the input of neural network model.

We can conclude based on the experiments and final results which creating profiles from the implicit feedback based data still can improve the recommendations performance. Also, Combining OLAP model to use the performance of OLAP engines on top of a predicted utility matrix can be a very nice idea which can let the company to have different more accurate recommendations for each customer based on different contexts.

Furthermore, using the distributed platform like Hadoop or Spark is not recommended all the time, since it can just bring the technical cost for the company, especially when based on data, updating or training a model frequently is not necessary. For example, for our problem, the number of hotels can just increase inside a city maybe after a few months even years and the number of companies as the customers cannot change frequently (even if we have a completely new user, we can recommended the hotels mapped in to the same price category of their flight class or just popular hotels in a city), then the model can update just one time per year, which proves this idea which using the distributed platform since the first step is not recommended all the time.

Future Works

One interesting work can be to build a model with contextualizing method to see the difference in result with the current model. Also, comparing the result with using the same model just with tensor factorization can be interesting. Anyway, there are a couple of things we can do to improve the results. First of all, for our neural network model, it's better to run the neural network on top of the final model (LightFm_Model4_WARP) to give the input to this model to have a baseline to compare the results. Furthermore, we can give a bag of words of all of the information related to each hotel and company without any selection (e.g we can give the name of all destination cities or all of rate codes which a company used), also we can add visual features as the embeddings in our model (e.g we can give the best chosen image per each hotel as its profile), maybe we do not have a better result even, but at least we can try to become sure that our final profiles here are the best chosen profiles. Then we can be sure that we have a model which gives us the best result by identifying the weights of different features in profiles.
Also, we can at least choose 5 companies per each city and 5 hotels which each of them at least has booked 5 times before. Then we can be sure our model can recommend at least five hotels per each city. If we will add more context in addition to city such as month, price and etc to our final model,we should consider this condition for those contexts as well.

One idea for the company can be collecting more information about companies and hotels from business travelers, then we have some external side information for users and

items which can see how much they can change the recommendation performance. This information can be gathered by asking the business travellers during booking a hotel of flight ticket, or it can be collected by sending some questionnaires to current customers and ask them to fill them in.

Furthermore, for evaluation, the best way will be user study or online methods which user tell us the recommended hotels from our model are good recommendations or not. However, since our users are from different companies, this method cannot be done easily but still we can just choose some companies which are more eager to help us.

Bibliography

[Adomavicius 2001] Gediminas Adomavicius and Alexander Tuzhilin. *Multidimensional recommender systems: a data warehousing approach.* In Electronic commerce, pages 180–192. Springer, 2001. (Cited on page 16.)

[Adomavicius 2005a] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen and Alexander Tuzhilin. *Incorporating contextual information in recommender systems using a multidimensional approach.* ACM Transactions on Information Systems (TOIS), vol. 23, no. 1, pages 103–145, 2005. (Cited on pages 12 and 16.)

[Adomavicius 2005b] Gediminas Adomavicius and Alexander Tuzhilin. *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, vol. 17, no. 6, pages 734–749, 2005. (Cited on pages 5 and 9.)

[Adomavicius 2015] Gediminas Adomavicius and Alexander Tuzhilin. *Context-aware recommender systems.* In Recommender systems handbook, pages 191–226. Springer, 2015. (Cited on pages 17 and 18.)

[Aligon 2013] Julien Aligon. *Similarity-based recommendation of OLAP sessions.* PhD thesis, Tours, 2013. (Cited on page 39.)

[AOULLAY 2017] Amine AOULLAY. *Cross Sell Performance Monitoring.* In Master thesis report. Amadeus, 2017. (Cited on page 20.)

[Arruza 2016] Michael Arruza, John Pericich and Michael Straka. *The Automated Travel Agent: Hotel Recommendations Using Machine Learning.* 2016. (Cited on page 20.)

[Bobadilla 2013] Jesús Bobadilla, Fernando Ortega, Antonio Hernando and Abraham Gutiérrez. *Recommender systems survey.* Knowledge-based systems, vol. 46, pages 109–132, 2013. (Cited on pages 10, 11, 12, 13 and 15.)

[Burke 2002] Robin Burke. *Hybrid recommender systems: Survey and experiments.* User modeling and user-adapted interaction, vol. 12, no. 4, pages 331–370, 2002. (Cited on page 13.)

[Candillier 2007] Laurent Candillier, Frank Meyer and Marc Boullé. *Comparing state-of-the-art collaborative filtering systems.* In International Workshop on Machine Learning and Data Mining in Pattern Recognition, pages 548–562. Springer, 2007. (Cited on page 9.)

[Davidson 2003] Rob Davidson and Beulah Cope. Business travel: Conferences, incentive travel, exhibitions, corporate hospitality and corporate travel. Pearson Education, 2003. (Cited on page 1.)

[Ekstrand 2011] Michael D Ekstrand, John T Riedl, Joseph A Konstan*et al. Collaborative filtering recommender systems.* Foundations and Trends® in Human–Computer Interaction, vol. 4, no. 2, pages 81–173, 2011. (Cited on pages 1, 9, 11, 12 and 13.)

[Fu 2016] Lixin Fu. *A Recommendation System Using OLAP Approach.* In Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on, pages 622–625. IEEE, 2016. (Cited on page 39.)

[Geurts 2006] Pierre Geurts, Damien Ernst and Louis Wehenkel. *Extremely randomized trees.* Machine learning, vol. 63, no. 1, pages 3–42, 2006. (Cited on page 63.)

[Herlocker 2000] Jonathan L Herlocker, Joseph A Konstan and John Riedl. *Explaining collaborative filtering recommendations.* In Proceedings of the 2000 ACM conference on Computer supported cooperative work, pages 241–250. ACM, 2000. (Cited on page 6.)

[Hu 2008] Yifan Hu, Yehuda Koren and Chris Volinsky. *Collaborative filtering for implicit feedback datasets.* In Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on, pages 263–272. Ieee, 2008. (Cited on pages 23 and 50.)

[Hug 2017] Nicolas Hug. *Surprise, a Python library for recommender systems.* 2017. (Cited on page 46.)

[Isinkaye 2015] FO Isinkaye, YO Folajimi and BA Ojokoh. *Recommendation systems: Principles, methods and evaluation.* Egyptian Informatics Journal, vol. 16, no. 3, pages 261–273, 2015. (Cited on pages 10, 12, 13, 14 and 15.)

[Kamakura 2008] Wagner A Kamakura. *Cross-selling: Offering the right product to the right customer at the right time.* Journal of Relationship Marketing, vol. 6, no. 3-4, pages 41–58, 2008. (Cited on page 1.)

[Koren 2009] Yehuda Koren, Robert Bell and Chris Volinsky. *Matrix factorization techniques for recommender systems.* Computer, vol. 42, no. 8, 2009. (Cited on page 13.)

[Koren 2015] Yehuda Koren and Robert Bell. *Advances in collaborative filtering.* In Recommender systems handbook, pages 77–118. Springer, 2015. (Cited on page 47.)

[Kotsiantis 2006] Sotiris Kotsiantis and Dimitris Kanellopoulos. *Association rules mining: A recent overview.* GESTS International Transactions on Computer Science and Engineering, vol. 32, no. 1, pages 71–82, 2006. (Cited on page 35.)

[Krishna 2017] Vaibhav Krishna and Nino Antulov-Fantulin. *Learning from Incomplete Ratings using Nonlinear Multi-layer Semi-Nonnegative Matrix Factorization.* arXiv preprint arXiv:1710.05613, 2017. (Cited on page 47.)

[Krohn-Grimberghe 2010] Artus Krohn-Grimberghe, Alexandros Nanopoulos and Lars Schmidt-Thieme. *Integrating OLAP and recommender systems: an evaluation perspective.* In Proceedings of the ACM 13th international workshop on Data warehousing and OLAP, pages 85–92. ACM, 2010. (Cited on page 39.)

[Kula 2015] Maciej Kula. *Metadata Embeddings for User and Item Cold-start Recommendations.* In Toine Bogers and Marijn Koolen, editeurs, Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located

with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015., volume 1448 of *CEUR Workshop Proceedings*, pages 14–21. CEUR-WS.org, 2015. (Cited on pages 51 and 57.)

[Li 2011] Shibo Li, Baohong Sun and Alan L Montgomery. *Cross-selling the right product to the right customer at the right time.* Journal of Marketing Research, vol. 48, no. 4, pages 683–700, 2011. (Cited on page 1.)

[Li 2017] Siqi Li. *Implicit Feedback Based Context-Aware Recommender For Music Light System.* 2017. (Cited on pages 21, 22, 49 and 50.)

[Lu 2015] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang and Guangquan Zhang. *Recommender system application developments: a survey.* Decision Support Systems, vol. 74, pages 12–32, 2015. (Cited on page 10.)

[Luo 2012] Xin Luo, Yunni Xia and Qingsheng Zhu. *Incremental collaborative filtering recommender based on regularized matrix factorization.* Knowledge-Based Systems, vol. 27, pages 271–280, 2012. (Cited on page 50.)

[Mavalankar 2016] Aditi A Mavalankar, Ajitesh Gupta, Chetan Gandotra and Rishabh Misra. *Hotel Recommendation System.* 2016. (Cited on page 9.)

[Mavalankar 2017] Aditi A Mavalankar, Ajitesh Gupta, Chetan Gandotra and Rishabh Misra. *Hotel Recommendation System.* 2017. (Cited on page 19.)

[Nilashi 2017] Mehrbakhsh Nilashi, Karamollah Bagherifard, Mohsen Rahmani and Vahid Rafe. *A recommender system for tourism industry using cluster ensemble and prediction machine learning techniques.* Computers & Industrial Engineering, vol. 109, pages 357–368, 2017. (Cited on page 9.)

[Rahman 2013] Mohammed Mahmudur Rahman. *Contextual recommendation system.* In Informatics, Electronics & Vision (ICIEV), 2013 International Conference on, pages 1–6. IEEE, 2013. (Cited on page 17.)

[Rajeswari 2015] K Rajeswari. *Feature Selection by Mining Optimized Association Rules based on Apriori Algorithm.* International Journal of Computer Applications, vol. 119, no. 20, 2015. (Cited on page 34.)

[Rendle 2009] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner and Lars Schmidt-Thieme. *BPR: Bayesian personalized ranking from implicit feedback.* In Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence, pages 452–461. AUAI Press, 2009. (Cited on page 48.)

[Schafer 2007] J Ben Schafer, Dan Frankowski, Jon Herlocker and Shilad Sen. *Collaborative filtering recommender systems.* In The adaptive web, pages 291–324. Springer, 2007. (Cited on pages 11 and 12.)

[Shahzad 2013] Waseem Shahzad, Salman Asad and Muhammad Asif Khan. *Feature subset selection using association rule mining and JRip classifier.* International Journal of Physical Sciences, vol. 8, no. 18, pages 885–896, 2013. (Cited on page 34.)

[Shani 2011] Guy Shani and Asela Gunawardana. *Evaluating recommendation systems.* In Recommender systems handbook, pages 257–297. Springer, 2011. (Cited on page 21.)

[Sharma 2013] Lalita Sharma and Anju Gera. *A survey of recommendation system: Research challenges.* International Journal of Engineering Trends and Technology (IJETT), vol. 4, no. 5, pages 1989–1992, 2013. (Cited on pages 14 and 15.)

[Shenoy 2017] Gourav G Shenoy, Mangirish A Wagle and Anwar Shaikh. *Kaggle Competition: Expedia Hotel Recommendations.* arXiv preprint arXiv:1703.02915, 2017. (Cited on pages 9 and 18.)

[Steeg 2015] F van Steeg. Context-aware recommender systems. Master's thesis, 2015. (Cited on pages 10 and 23.)

[Vincent 2010] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio and Pierre-Antoine Manzagol. *Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.* Journal of machine learning research, vol. 11, no. Dec, pages 3371–3408, 2010. (Cited on page 59.)

[Wang 2015] Hao Wang, Naiyan Wang and Dit-Yan Yeung. *Collaborative deep learning for recommender systems.* In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 1235–1244. ACM, 2015. (Cited on pages 58, 59 and 68.)

[Weston 2011] Jason Weston, Samy Bengio and Nicolas Usunier. *Wsabie: Scaling up to large vocabulary image annotation.* In IJCAI, volume 11, pages 2764–2770, 2011. (Cited on pages 49 and 62.)

[Weston 2012] Jason Weston and John Blitzer. *Latent structured ranking.* arXiv preprint arXiv:1210.4914, 2012. (Cited on page 12.)

[Xie 2009] Jianwen Xie, Jianhua Wu and Qingquan Qian. *Feature selection algorithm based on association rules mining method.* In Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on, pages 357–362. IEEE, 2009. (Cited on pages 33 and 34.)

[Yang 2018] Longqi Yang, Eugene Bagdasaryan, Joshua Gruenstein, Cheng-Kang Hsieh and Deborah Estrin. *OpenRec: A Modular Framework for Extensible and Adaptable Recommendation Algorithms.* In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. ACM, 2018. (Cited on pages 58, 59 and 68.)

[Yao 2015] Guanwen Yao and Lifeng Cai. *User-Based and Item-Based Collaborative Filtering Recommendation Algorithms Design.* 2015. (Cited on page 9.)

Abstract: In real world, mostly the ratings from users are not provided. Implicit feedback from historical booking data can be the only available data. This data is often very sparse, causing collaborative filtering techniques to decrease significantly in their recommendation performance. To solve the sparsity problem, side information (e.g. item or user profiles) can be utilized. Also, Matrix Factorization (MF) is an appealing recent method which can handle the sparsity problem, the latent user or item representations learned by gradient descent methods in MF can improve the recommendation performance. Sometimes, having access to the side information cannot be provided. One idea can be creating these profiles by extracting information from implicit data. In this master thesis, we built a hotel recommendation system for corporate travellers based on historical booking data. It is shown that we can improve the performance by using this kind of profiles as the content information and join them with collaborative filtering for the feedback information. Furthermore, post-filtering on aggregation of different contexts for having more accurate result can be applied to the model. Considering the OLAP architecture integrated by our final result from our model, we can contextualize them on different aggregation on contexts.
All this research work has been implemented in python version 3.6.
Keywords: Implicit feedback, Matrix Factorization,profile embeddings, Context_aware, OLAP integration, Recommendation Systems