An intelligent hybrid multi-criteria hotel recommender system using explicit and implicit feedbacks

Ashkan Ebadi

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Computer Science

Concordia University

Montréal, Québec, Canada

April 2016

CONCORDIA UNIVERSITY
SCHOOL OF GRADUATE STUDIES

This is to certify that the thesis prepared

By: Ashkan Ebadi

Entitled: An intelligent hybrid multi-criteria hotel recommender system using explicit and implicit feedbacks

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

Dr. Yuhong Yan — Chair

Dr. Thomas Fevens — Examiner

Dr. Olga Ormandjieva — Examiner

Dr. Adam Krzyzak — Supervisor

Approved by

Chair of Department or Graduate Program Director

Dean of Faculty

Date

ABSTRACT

An intelligent hybrid multi-criteria hotel recommender system using explicit and implicit feedbacks

Ashkan Ebadi
Concordia University, 2016

Recommender systems, also known as recommender engines, have become an important research area and are now being applied in various fields. In addition, the techniques behind the recommender systems have been improved over the time. In general, such systems help users to find their required products or services (*e.g.* books, music) through analyzing and aggregating other users' activities and behavior, mainly in form of reviews, and making the best recommendations. The recommendations can facilitate user's decision making process. Despite wide literature on the topic, using multiple data sources of different types as the input has not been widely studied. Recommender systems can benefit from the high availability of digital data to collect the input data of different types which implicitly or explicitly help the system to improve its accuracy. Moreover, most of the existing research in this area is based on single rating measures in which a single rating is used to link users to items.

This dissertation aims to design a highly accurate hotel recommender system, implemented in various layers and tailored for the subject problem. Using multi-rating system and benefitting from large-scale data of different types, the recommender system suggests hotels that are personalized and tailored for the given user. The system employs natural language processing techniques to assess the sentiment of the users' reviews and extract implicit features. The entire recommender engine contains multiple sub-systems, namely users clustering, matrix factorization module, and hybrid recommender system. Each sub-system contributes to the final composite set of recommendations through covering a specific aspect of the problem. The accuracy of the proposed recommender system has been tested intensively where the results confirm the high performance of the system.

ACKNOWLEDGMENT

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Adam Krzyzak, for his continuous support, valuable guidance, and consistent encouragement throughout my research. This work was possible only because of your unconditional support, understanding, and immense knowledge. Thank you for providing me with the opportunity to learn. I would also like to thank my committee members for the brilliant comments and insightful suggestions. This last word of acknowledgment I have saved for my wonderful family for their endless support, encouragement, and love.

Contents

List of Figures

List of Tables

1. INTRODUCTION

Recent progress in information technology has provided us with various sources of data about almost everything. Although the availability of large-scale data can be beneficial, it can also make the decision making process more difficult. Users and customers have a lot of options to choose from which might make them confused in selecting the best possible and/or the most suitable item. In this sense, it is important to filter the information and personalize it for the use of each specific user. Recommender systems are one of the means for making personalized suggestions of items to the users based on their needs and preferences.

Nowadays, recommender systems are being widely used in different services covering vast area of applications. Following the boom in tourism industry and data technology during the past decade, the travel recommender systems have attracted considerable attention of researchers. As a tourist, most of the times, it is really confusing to decide where to go and to select among a large number of possible destinations, especially for unseen and unfamiliar places (Ricci, 2002). Hence, information retrieval and decision support systems are widely recognized as valuable tools in this context. In this respect, tourism and travel recommender systems have become a hot topic recently and attracted the attention of both researchers and companies.

However, most of the existing recommender systems in tourism employ a simple method which, in general, compares the profile of a given tourist with certain features of the available items (*e.g.* destination) and use them to predict the tourist's preferences (Ricci 2002; Gavalas & Kenteris, 2011). This is especially true about mobile recommender systems (Yang & Hwang, 2013). In such systems, a given tourist, *i.e.* the user, is asked to provide the system with a set of parameters that represent his/her interests, needs or limitations which are used by the system to make the recommendation by correlating the user's responses with the available destinations/packages. These methods are also called content-based recommendations (Gavalas & Kenteris, 2011).

Another approach is obtaining useful information from other tourists who have common or similar interests to the given user. In addition, systems can also benefit from the fact that travelers who are in close proximity might share common needs or interests (de Spindler *et al.*, 2006). Despite the recent advances in travel recommender systems, most existing recommender

systems have been unsuccessful in exploiting the information, reviews, or ratings that are being provided by similar tourists (Yang & Hwang, 2013).

In this thesis, an intelligent hybrid travel recommender solution is proposed. The proposed recommender engine is based on both the content data and similarities among users, exploiting implicit and explicit users' feedbacks. In addition, using data sources of different types, it employs multi-criteria rating approach to better capture users' preferences and augment the accuracy of the recommendations. The system is designed in different layers, using multiple sub-recommender systems each addressing specific aspect of the subject problem, aiming to increase efficiency and effectiveness of recommendations by considering. The proposed system is trained with TripAdvisor data collected from multiple sources and integrated into a single database. The final solution is verified and tested in different settings and scenarios to confirm and validate its accuracy.

The remainder of the document is organized as follows. The next chapter presents the background of the subject and reviews the respective literature. In Chapter 3, the objectives of the research are stated while Chapter 4 discusses data and methodologies used for the analysis. Chapter 5 reports the evaluation results while the conclusions are presented in Chapter 6. Chapter 7 suggests some directions for the future research and addresses the limitations.

2. LITERATURE REVIEW

The first papers on collaborative filtering recommenders were published in mid 1990s (*e.g.* Resnick *et al.*, 1994; Shardanand & Maes, 1995). Since then recommender systems have always been an important research field in machine learning. Recommender systems can be regarded as tools that directly help users to find their required products, services or contents (*e.g.* book, music, movie, web site, etc.) through aggregating and analyzing the activities and suggestions obtained from other users in terms of reviews and ratings (Frias-Martinez *et al.*, 2006; Frias-Martinez, Chen, & Liu, 2009). In general, such systems are created on computed probabilities for users versus items or the similarity between users.

In a broader context, recommender systems are categorized into content-based filtering (*CB*) and collaborative filtering (*CF*). CF systems employ information filtering techniques on users' purchase history or users' previous reviews/ratings on the items. Although this technique has been widely used in various applications, it suffers from at least two major issues: 1) sparsity, and 2) scalability (Claypool *et al.*, 1999; Sarwar *et al.*, 2000). In CB method, the contents of the user's reviews are being analyzed to infer the user profile which can be used to predict and recommend further items (Basu, Hirsh, & Cohen, 1998). However, CB recommender systems mainly propose items that are very similar to what user has bought before or the user is well aware of. The reason is that such method measures similarities between items which share same features or characteristics (Blanco-Fernández *et al.*, 2008). The relevant literature is reviewed in more detail in the rest of this section. In addition, a number of applications of recommender systems are discussed.

2.1 Recommender Systems, Applications and Examples

Recommender systems have several applications and are being used in a wide variety of fields and contexts ranging from online shopping and social networks to even intelligent health solutions. Although they were traditionally introduced in the World Wide Web applications, we can now see them even on mobile devices. Therefore, due to the availability of large-scale data, the use of recommender systems in data-intensive fields or applications where a given user should choose from a variety of options seems inevitable. Such systems facilitate the decision making process by providing the users/customers with personalized items/products which fit

their expectations and preferences. Moreover, recommender systems can provide users with a wide range of items to compare which will result in a better decision. They can also provide the user with the ability to explore the products, discover the interesting items (according to the given user), through a personalized recommendation system. Due to the mentioned reasons, recommender systems have become extremely common especially in recent years.

Four common functionalities can be considered for recommender systems: 1) To predict rating that a user *u* might give to an item *i*. This is one of the most studied areas in recommender systems literature, which attracted the attention of researchers especially right after the Netflix challenge (Bennett & Lanning, 2007). 2) To predict the ordering of items according to a measure, this is called the ranking prediction problem. In this context, the system focuses on the fact that the items with the highest predicted ratings are not necessarily the best items for a user. Thus, system distinguishes among the items that the user likes and the items that the user already know about them (Herlocker *et al.*, 2004). 3) Contextual non-personalized item-to-item recommendation in which for an item *i* a list of similar items is generated. This is being widely used by Amazon. 4) Personalized recommendation of items, which also called item-based top-*n* recommendation (Karypis, 2001; Deshpande & Karypis, 2004).

Some of the most popular applications of recommender systems are found in recommending commercial products (Schafer, Konstan, & Riedl, 2001) such as movies, music, books, or suggesting a research paper or partner, and even friend suggestion in social media. In the rest of this section, Amazon, a well-established commercial recommender system, Lifetrak, a well-documented academic project developed at the University of California, and Last.fm a social music platform are discussed in detail.

2.1.1 Amazon

Amazon[1], an American e-commerce and cloud computing giant, is one of the biggest online stores in the entire world and is the largest online retailer in the United States (Jopson, 2011). Started as an online bookstore in 1994, it now sells a wide variety of goods such as books, CDs, DVDs, MP3 downloads and streaming, *etc*. Recently, they offered new and slightly

[1] http://www.amazon.com

different categories of products such as electronics, furniture and even food. Meanwhile, it is ranked as one of the world's largest providers of cloud infrastructure services.

Amazon has one of the most famous recommender systems which is mostly used as an example of a recommender system to non-technical people. The algorithm is mainly based on item-based collaborative filtering and it tracks the items that a customer has bought before making similar recommendations. Customers can rate back their purchases using a 5-star rating scale, which helps the recommender system to improve its recommendations. More interestingly, Amazon follows the products that a customer is currently looking for to propose complementary items (named *better together*) or suggest other items that previous customers have bought in combination with the target product (called *customers who bought this also bought*).

2.1.2 Lifetrak

Lifetrak is a music player which provides the user with a context-aware music experience (Reddy & Mascia, 2006). The context-aware music engine is influenced by several factors such as user's location, time of operation, and the environment information like weather and traffic. The system is also using users' feedbacks on the recommended songs to assess whether the suggested music was appropriate for that particular context. The song rating is relatively simple and uses some forms of content-based approach. Lifetrak was designed for mobile devices.

2.1.3 Last.fm

Last.fm[2], launched in 2002, is a British music platform which uses a music recommender system called *Audioscrobbler*. Audioscrobbler is a personalized recommender system which tracks the musical interests of the users. The user can explicitly rate music tracks in a binary-scale rating, *i.e.* love it or hate it. In addition, an implicit rating approach is also implemented which takes the listening time as a measure of satisfaction and uses it to improve the recommendations. Audioscrobbler is also based on collaborative filtering algorithm through creating a detailed record for each user. It not only analyzes the songs that a user has listened to but also uses its huge social network data to make the recommendations.

2.2 Recommender Systems, Requirements

[2] http://www.last.fm

In order to have an operational and functional recommender system, a number of requirements or mandatory operational functions should be considered. In this section, such requirements are briefly discussed.

2.2.1 Multiple Data Sources

A highly operational recommender system should be able to use multiple data sources, since in most of the cases, especially in industrial applications, the designed system might be used in different contexts. For example, it might be working with or without accessing the items' metadata. Another reason is due to the evolution and change of the context over time. This ability and flexibility of the system can be also useful for overcoming the cold-start problem (Burke, 2007), which will be discussed in the next section.

2.2.2 The Cold-Start Problem

The cold-start problem refers to the beginning stage of launching the recommendation system within which the system might not be able to provide useful recommendations, mainly due to the lack of enough data (Su & Khoshgoftar, 2009). Thus, an operational recommender system should be able to manage the cold-start problem. Several approaches are available for dealing with the cold-start problem. One of the approaches for managing the cold-start problem is using metadata or socio-demographic data to infer the profiles (Nguyen, Denos, & Berrut, 2007). Another way would be employing external data, *e.g.* other usages of the items that are better known, to make the recommendations (Poirier, Fessant, & Tellier, 2010). The external data can be used until the system gathers enough data about the objects. Another related problem is about new items or new users for which the system again does not have enough data (Adomavicius & Tuzhilin, 2005).

2.2.3 Robustness

The recommender system is needed to be robust against noisy or corrupted data. This becomes even more important if the system uses multiple data sources. For example, if in a collaborative recommender, rating logs are corrupted it can highly affect the performance of the system. Thus, the robustness is an important issue which is linked with the system performance (Lam & Riedl, 2004), and some measurements should be taken before using the data. Another

example is in social networks where a lot of fake users might exist, hence, their profiles and ratings should be filtered out before applying the recommendation process.

2.2.4 Scalability

Like all the computer systems, scalability is a key factor for recommenders. The system should have been designed to work efficiently and effectively, even if the number of users or items increase unexpectedly (Takács *et al.*, 2009). That is the quality of systems and its performance should not suffer from increased number of users or items.

2.2.5 Transparency

The recommender system design should be easy to explain and a confidence index should be provided for its recommendation (Basu, Hirsh, & Cohen, 1998). This plays an important role for a recommender system to be accepted by the users (Bilgic, Mooney, & Rich, 2004).

2.3 Recommender Systems Topologies

Several topologies exist for recommender system and a number of classifications of recommender systems can be found in the literature. In this section, three main categories of recommender systems are discussed.

2.3.1 Classical Categorization

According to the classical categorization, recommender systems can be divided into three types as: 1) collaborative filtering, 2) content-based filtering, and 3) the hybrid systems. This classification has been widely used in the literature (*e.g.* Adomavicius & Tuzhilin, 2005). This classification is mainly based on the type of the input which is used in recommender systems. In collaborative filtering, users' logs along with users' ratings on items are used. In content-based filtering items' metadata are mainly used. In collaborative filtering users' logs are compared based on some similarity measure, where in content-based filtering the metadata are compared to find users' preferences on items. Hybrid recommenders are a combination of the both mentioned methods. These methods will be discussed in more detail in Section 2.4.

2.3.2 Su and Khoshgoftar Categorization

Su and Khoshgoftar (2009) proposed a classification of purely collaborative systems, thus it can be considered as a sub-classification of recommender systems which is limited to the collaborative methods and hybrid techniques. They classified the collaborative systems into three sub-categories: 1) memory-based collaborative techniques which are mainly based on the nearest-neighbors algorithm, 2) model-based collaborative systems which covers a wide range of techniques such as clustering, matrix factorization, Bayesian networks, and Markov decision process, and 3) hybrid collaborative filtering systems in which a collaborative filtering approach is combined with other collaborative methods.

2.3.3 Rao Categorization

Rao (2010) focuses on the source of information which is used in the recommender system and proposes another classification of recommender systems as follows: 1) content-based filtering methods in which the system uses correlations among items, or matches item features with users' profiles to make the recommendation, 2) collaborative filtering which focuses on users and the items that they have used, 3) demographic filtering in which a priori knowledge is available on the groups of users and can be used to build stereotypes to be linked with the list of items, 4) hybrid filtering methods which are a combination of collaborative filtering and content-based filtering, 5) knowledge-based recommender systems where a priori knowledge about the relationships between users and items are available, and 6) utility-based recommenders in which a utility measure is calculated for a user and an item.

2.4 Recommender Systems Approaches

The root of modern recommendation techniques is traced back to information filtering where the irrelevant data to a given user is filtered out (Huang, Chung, & Chen, 2004; Kantor *et al.*, 2011). Different algorithms have been used for designing and implementing the recommender systems. Each approach has its own advantages and drawbacks. In this section, the state-of-the-art is presented and some common approaches which were briefly presented in the previous section are discussed in detail.

2.4.1 Collaborative Filtering

Collaborative filtering is based on the nature of humans, that is people share ideas and opinions and explicitly or implicitly influence each other's decisions. In collaborative filtering, the opinions of other (similar) people are evaluated to infer the required information (Schafer *et al.*, 2007). For this purpose, the interests of users are predicted through collecting related information from a large number of similar users. This method is based on the assumption that those people who agreed in the past will more likely continue to agree in the future. In real-life problems (*e.g.* e-commerce), systems that are based on the collaborative filtering method should process a huge amount of information which apart from being resource-consuming, it might also harm the systems' responsiveness. However, since collaborative filtering method has been in the core of researchers' attention during the last decade, it is now considered as one of the prominent personalization techniques in the field.

As one of the fundamental approaches to recommendation, collaborative filtering has several forms. In user-based collaborative filtering, the recommendation is made to a user through finding other similar users and suggesting the things that they like (Resnick *et al.*, 1994; Herlocker *et al.*, 1999). In another approach, named as item-based collaborative filtering, the system finds and recommends similar items to those the user has already bought or liked (Sarwar *et al.*, 2001; Deshpande & Karypis, 2004). Hence, item-based algorithms can be considered as the transpose of the user-based ones.

To find the similarity between users, a user-item preference matrix should be created. Such matrix can be of huge size in complex problems. Matrix factorization methods take the user-item matrix and decompose it into a more compact representation which is used to predict the preference of the items that a given user has not seen yet or to prioritize among a set of seen items. One of the common techniques in matrix factorization is called singular value decomposition or SVD (Deerwester *et al.*, 1990; Sarwar *et al.*, 2002). Gradient descent is mainly used for factorizing the matrix in a computationally efficient way; however, it does not contain all the properties of proper singular value decomposition (Funk, 2006). Other common techniques include factor analysis (Canny, 2002) and eigenvalue decomposition (Goldberg, 2001). After creating the preference matrix, collaborative filtering method calculates a similarity measure between the given user's profile and the other users. Various distance-based or

correlation-based metrics can be used to calculate the profile similarity measure, *e.g.* Euclidean distance, Pearson correlation and cosine similarity to name the few. The final output will be a single item or a list of items (top *n* recommended items) which the given user will most likely be interested (Cosley, 2003).

Collaborative filtering techniques suffer from two main problems, namely sparsity and the first-rater problem. Sparsity problem arises as most users only rate a very small portion of the available items, making it very difficult to find similar users. The second problem is caused by the fact that an item cannot be recommended unless it has received a user's rating. A special case would be newly added items which will be biased by such recommendation method (Melville, Mooney, & Nagarajan, 2001). Despite the mentioned drawbacks, collaborative filtering can provide the user with fitting recommendations which do not share almost anything with the items previously rated by the user. Thus, this method has been proven successful in a wide variety of domains and applications (McLaughlin & Herlocker, 2004).

2.4.2 Content-Based Filtering

Content-based filtering is another common approach for designing recommender systems which are based on items' descriptions and users' preferences (Brusilovsky, Kobsa, & Nejdl, 2007). Such systems mainly use a set of keywords to describe the items while in parallel a profile is built for users indicating the type of items they prefer. Hence, content-based filtering recommends items to a given user similar to the ones the user liked in the past. Content-based methods have been proven to perform well in text-intensive domains/applications where there is enough content associated with the items. However, this approach requires significant knowledge engineering efforts to provide/collect the needed metadata about the items (Melville, Mooney, & Nagarajan, 2001, Cosley, 2003; O'Donovan & Smyth, 2005). In content-based systems, users' profiles are made independently from each other. For example, even if two categories of items are frequently agreed together by users, the system will never recommend items from a category if the user has not rated any items from that category. Therefore, a critical issue with content-based filtering is their ability to learn user preferences from users' actions and behavior.

2.4.3 Knowledge-Based Recommender Systems

Knowledge-based recommender systems incorporate inferences about users' preferences and needs to make the recommendation. The knowledge may sometimes contain explicit information about how certain characteristics of a product meets user's requirements (Felfernig & Burke, 2008). In other words, such systems are based on the knowledge about user preferences, item groupings, and recommendation criteria and are applied in situations where other approaches (*e.g.* collaborative or content-based filtering) cannot be employed.

2.4.4 Demographic Recommender Systems

This type of recommender systems employs demographic profile of users/items for making recommendations (Krulwich, 1997; Pazzani, 1999). Collecting the accurate data in large-scale is one of the issues in the mentioned recommender systems where data are mainly collected and integrated manually from the users or from the product catalogues. Sometimes, such systems also employ a pre-generated demographic clustering of users/items (Vozalis & Margaritis, 2004).

2.4.5 Hybrid Recommender Systems

In hybrid recommender systems, two or more different recommender algorithms/techniques/approaches are combined, creating a composite recommender (Burke, 2002). The main purpose in applications is to use hybrid recommenders (of various types) which outperform the individual algorithms (*e.g.* Torres *et al.*, 2004). One of the most important advantages of the hybrid recommender systems is the inclusion of algorithms that cover different aspects of the data and the subject problem in order to produce improved recommendations. For example, as seen before, one of the limitations of the item-based recommender systems is when there is no rating for an item. A hybrid recommender system can use text similarity measures to relate the new item with the existing items and increase the accuracy of the collaborative filtering method as users start rating the item.

Combining collaborative and content-based filtering methods in a hybrid framework has been considered by researchers recently where various approaches have been applied. Some examples are: using collaborative and content-based filtering separately and then combining their predictions, using collaborative filtering as the basis and adding content-based capabilities to the system, or incorporating both approaches in one model (Adomavicius & Tuzhilin, 2005). Netflix,

an American provider of Internet streaming media, is a famous example of the success of hybrid systems (Bennett & Lanning, 2007). Netflix analyzes the searching and watching preferences and habits of similar users (collaborative filtering) and recommends movies which have similar features as the ones the user has already highly rated (content-based filtering).

There exist a variety of studies combining various techniques to design a hybrid recommender system for different applications (*e.g.* Balabanović & Shoham, 1997; Melville, Mooney, & Nagarajan, 2001; Claypool *et al.*, 1999). For example, Melville and his colleagues proposed a hybrid system, named content-boosted collaborative filtering (Melville, Mooney, & Nagarajan, 2001). They first employ content-based predictions and convert a sparse user rating matrix to a full user rating matrix. Collaborative filtering is then applied on the created matrix to make the recommendations. Their results show that the performance of the content-boosted collaborative filtering overcomes the recommender systems which use either content-based or collaborative filtering approaches.

According to Burke (2002), hybrid recommender systems can be divided into seven categories:

1. *Weighted* systems, in which the scores are generated by a variety of recommenders and are combined to make the final recommendation(s) to the users.
2. *Switching* recommenders, in which the system switches between different algorithms to choose the best possible solution in a particular context.
3. *Mixed* recommenders. They are slightly similar to the weighted systems and present the results of a number of different recommenders. The difference is mixed recommenders do not necessarily combine the results into a single list of recommendations.
4. *Feature-combining* systems, in which several data sources are employed as inputs to a single recommender algorithm.
5. *Cascading* recommenders, where a chain of several (mostly simple) different algorithms is used, feeding the output of a recommender to the input of the other one.
6. *Feature-augmenting* recommenders, in which the output of an algorithm is used as one of the input sources (features) of another algorithm.
7. *Meta-level* recommenders, which use an algorithm to train a model and then feed that model as an input to another algorithm.

Although hybrid recommender systems can often improve the accuracy, sometimes they incur considerable computational costs (Amatriain & Basilico, 2012). They are also useful for offering personalized recommendations by taking the requirements of different users or characteristics of different items into the account. In addition, in more complicated frameworks/applications, hybrid systems can be also implemented through integrating feature-based methods with collaborative filtering and forming a single learning model, which is often in the form of a matrix factorization model (*e.g.* Shan & Banerjee, 2010; Chen *et al.*, 2012).

2.5 Recommender Systems - Data Sources and Targets

In general, four basic objects are used in recommender systems, namely users, items, users' features such as age (range), and items' metadata, where data can be about any meaningful two pairs of the mentioned objects, in any meaningful format such as Boolean or an integer. For example, if we consider users and items objects, the created matrix can contain users' ratings on different items. Another example can be items and their metadata, which might provide us with an array of the catalog items. In this thesis, an example of item and metadata would be the case of users' reviews about hotels and the extracted keywords (metadata) out of them. One should note that in most of the applications binary data sources are preferred, and the other data sources are sometimes converted to binary values (if required), based on the problem characteristics. Based on the objective of the recommender system, the target might differ. However, most of the recommender systems mainly focus on recommending items to users. There exist other targets for recommender systems, such as metadata or socio-demographic recommendations.

2.6 Recommender Systems - Common Algorithms

In this section a number of common algorithms that are widely used in recommender systems are discussed. Among those k-Nearest-Neighbors and Matrix Factorization are ones of the most common the methods which have been used in both academia and industry.

2.6.1 Association Rules

Association rules are a well-known technique in data mining, *e.g.* the Apriori algorithm (Agrawal & Srikant, 1994). Such techniques learn from the data and extract rules which predict

the occurrence of an item based on the other items' occurrences. There are also a number of studies which employed association rules in the context of recommender systems (*e.g.* Mobasher *et al.*, 2001; Lin, Alvarez, & Ruiz, 2002). However, association rules need to be adopted according to the application of the recommender system. That is in recommender systems any item can be recommended, thus the association rules should be able to capture the associations among any items, even if the support is small. Here, it becomes challenging since setting a small threshold for the support can lead to a large set of associations! There exist some heuristic approaches, such as adaptive support (Mobasher *et al.*, 2001) and sliding windows (Davidson *et al.*, 2010), to overcome the mentioned problem.

2.6.2 Bayesian Classifiers

In Bayesian network classifiers all features are considered as random continuous or discrete variables (Friedman, Geiger, & Goldszmidt, 1997). In particular, Bayes theorem and conditional probabilities are used in Bayesian classifiers to classify the given data through maximizing the posterior probability of the items' class. In recommenders' context, ratings can be considered as classes and the Bayesian classifier can be applied on the real-valued ratings. It is assumed that given a class (*e.g.* rating), users (or items) are independent, and thus the probability of the class is calculated. However, it is obvious that such assumption is not satisfied in collaborative filtering where it is assumed that users and/or items are related. For this reason, Bayesian classifiers are mostly coupled with another algorithm in recommender systems. For example, Candillier, Meyer, and Boullé (2007) coupled a Bayesian model with a k-Nearest-Neighbors algorithm.

2.6.3 Neural Networks

Neural Networks (NN) are widely used in computer science in a variety of applications (Bishop, 2006). Despite the large coverage of NN algorithms in machine learning, they are not so common in recommender systems due to several reasons such as their low convergence speed in high dimensional problems, or the fact that most of the time there is no need to employ NN as complex non-linear classifiers in recommenders (Pazzani & Billsus, 1997). Another drawback is about the black box effect of NN algorithms such that the output of the system might be hard to

interpret. However, some instances of NN algorithms were successfully used in recommender systems, *e.g.* fast matrix factorization techniques (Takács *et al.*, 2008).

2.6.4 K-Nearest Neighbors

K-Nearest-Neighbor (KNN) based algorithms, also called memory-based algorithms, are widely used in the context of recommender systems (Su & Khoshgoftar, 2009). They can be considered as a generalization to the association rules as they go over all the items and/or users in the corpus. One of the serious limitations KNN approaches is their lack of scalability. Moreover, they might be time consuming in large-scale real-life applications as the time needed for building the model is quadratic, *i.e.* a function of squared number of objects in the corpus.

2.6.5 Matrix Factorization

It was in Netflix challenge when matrix factorization techniques became very popular (Bell & Koren, 2007). Matrix factorization algorithms are not only fast and accurate but also relatively easy to implement. However, they might be difficult to be adopted for item-item recommendations. In general, such techniques transform a given matrix into typically three simpler matrices. In recommender systems, matrix factorization techniques should deal with the missing values problem. The first matrix factorization approaches handled the missing value problem through replacing the missing values of the rating matrix (Sarwar *et al.*, 2000), which was not an effective way since it resulted in large dense matrices. Another more effective approach is to use parameters and regularization (Takács *et al.*, 2008).

2.6.6 Other Algorithms

There exists a wide range of machine learning algorithms and techniques that are being used in data mining projects and recommender systems. For example, Latent Semantic Analysis (Deerwester *et al.*, 1990) is closely related to matrix factorization technique and is used for document representation. Other techniques that are worth mentioning include: Markov Decision Process (Shani, Brafman, & Heckerman, 2002), Boltzmann machines (Salakhutdinov, Mnih, & Hinton, 2002), and random walk systems (Jamali & Ester, 2009).

2.7 Recommender Systems - Evaluation

As discussed, there exist various approaches/algorithms towards designing the recommendation systems. Moreover, recommendation systems are being used in a wide area of applications/tasks. Hence, it is crucial to have some measures in hand to evaluate different techniques and designs in order to select the best possible solution for a given problem. Therefore, the final goal of the evaluation procedure would be to determine a recommender system's capability and ability in a given context to meet the main objectives. This can be a difficult task since trying different algorithms to see which one is performing better on a given dataset can be very costly. In this section, several evaluation methods are presented and discussed.

2.7.1 Offline Evaluation

Offline evaluation method has been widely used for analyzing recommender systems (*e.g.* Breese, Heckerman, & Kadie, 1998; Gunawardana & Shani, 2009). Although offline evaluations suffer from some limitations, they are mostly used as a single measure of evaluation. In addition, offline evaluation is mainly used to evaluate a target algorithm behavior before its final deployment or to select a set of algorithms/designs from a larger set of the potential designs, to be tested by users. However, it is better to determine the best performing algorithms for a given problem prior to having them tested by users since user trials might be an expensive procedure.

Using separate training and test datasets in a cross validation module is a basic and common approach in offline evaluation. The data is split into two disjoint sets, namely training and test datasets, and a recommender model is built on the training dataset. Then, the test set is also split into two parts, named query and target sets. The query set is used by the recommender system to predict ratings for the items in the target set or to recommend items where the performance of the system is evaluated. The whole process is performed as a part of a *k*-fold cross validation module where the data is divided into *k* equal sets named as folds. Each fold is used once as the test set while the other folds are considered as the training set within an iterative process. The overall performance of the recommender system is then evaluated through aggregating the results of each run. This approach is useful in mitigating the effects of undesired variations in the test set (Gunawardana & Shani, 2009).

Various metrics can be used in offline evaluation approach for measuring the performance of the recommender systems. For example, mean absolute error (*e.g.* Shardanand & Maes, 1995; Herlocker, Konstan, & Riedl, 2002), or root mean squared error (*e.g.* Herlocker et al., 2004; Bennett & Lanning, 2007) are used as accuracy proxies. The accuracy proxies are good where explicit users' ratings are available which makes it possible to compare the predicted rating with the exact values. Some other metrics such as precision/recall (Salton, 1992), F1 (Van Rijsbergen, 1979), ROC curves (Swets, 1963) can be also used to analyze the ability of the recommender system in distinguishing between relevant and irrelevant items. There exist other measures, *e.g.* mean reciprocal rank, that check how good is the recommender system in including at least one relevant item near the top of the list of recommendations. Discounted cumulative gain (Järvelin & Kekäläinen, 2002) is another similar measure which evaluates how good the recommender is in putting relevant items at the top and irrelevant items at the bottom of the list.

2.7.2 User-Based Evaluation

Offline evaluation method is relatively easy to implement and effective approach for examining recommender systems performance. However, one can make sure about the exact accuracy of a designed recommender system when it starts being used by real users of the system. If the recommender system succeeds to satisfy its users' requirements, it will be highly rated. Hence, in some applications, it is crucial to test the recommender systems with real users, which is called user-based evaluation.

The tests might include field trials in which users are provided with the real recommender system and their satisfaction is measured against the proposed new system. A/B testing is one of the common approaches in field trials where users are split into two different sets, one set is being provided with the new recommender system while the other set uses the existing old approach, and then users' satisfaction are compared. Apart from the field trials, another approach is performing qualitative statistical analysis. For example, surveys can be used to evaluate customers' satisfaction. Qualitative analysis is quite common in evaluating the usefulness of a proposed system (*e.g.* McNee *et al.*, 2002), or to asses users' interactions with a particular system (*e.g.* Bollen *et al.*, 2010).

2.7.3 More Discussion on Evaluation

Research on recommender systems mainly focuses on accuracy as a measure of higher performance (Herlocker *et al.*, 2004). However, other aspects should be also taken into consideration while evaluating the recommender systems. In other words, accurate recommendations are not necessarily the most useful recommendations. In an interesting paper, McNee and his colleagues argue that the recommender systems community should go beyond the traditional accuracy measures and propose new user-centric directions for evaluating the performance of recommender systems (McNee, Riedl, & Konstan, 2006). Moreover, different algorithms with the same accuracy level might influence users and their satisfaction differently (McNee *et al.*, 2002; Torres *et al.*, 2004).

Recently, several studies have focused on non-accuracy metrics for evaluating the performance of recommender systems (*e.g.* Ziegler *et al.*, 2005; Vargas & Castells, 2011; Willemsen *et al.*, 2014). One of the factors that is being considered in these new methods is diversity of recommendations, meaning that if the user is provided with diverse sets of items to select from. Diversity has been already recognized in information retrieval (Carbonell & Goldstein, 1998; Clarke *et al.*, 2008). Apart from diversity, novelty of recommendations has also attracted the attention of researchers (Vargas & Castells, 2011; Zhang & Hurley, 2008). Novelty focuses on the ability of the recommender system to suggest items that are completely new to the user. This idea is highly useful in entertainment industry (*e.g.* movies) where the aim is to suggest new unseen items to the user which turn out to be good. Other measures exist which target the stability of the recommender system in terms of recommendation rate changes over time (Burke, 2002; Adomavicius & Zhang, 2012). This measure deals with a trade-off since although stable recommender system might be more robust against unexpected changes (Lam & Riedl, 2004), the recommendations might be very predictable.

2.8 Travel Recommender Systems

Tourism is now considered as one of the largest industries in the world. According to the World Travel and Tourism Council (WTTC)[3], the share of tourism industry of the global Gross

[3] http://www.wttc.org

Domestic Product (GDP)[4] will rise from 9.1% in 2011 to 9.6% in 2021. Nowadays, people can travel easier than before thanks to the advances in the overall living standards as well as progress in information technology which provides travelers with fitted and tailored information. A considerable number of travel agents and companies, *e.g.* TripAdvisor[5] and Expedia[6], are now providing online services to the potential tourists.

Although the booming tourism industry along with the rapid growth of online travel data have provided tourists with new opportunities and have made the travel easier, the availability of large-scale (digital) travel data and the diversity of the service providers have imposed new challenges to the travelers. As a tourist, it is sometimes very difficult to choose from a large number of available travel options. Moreover, the travel companies have also faced with new issues due to the tighter competition and the need to provide the customers with the best fitted package. Thus, the demand for highly intelligent travel recommender systems is increasing.

Recommender systems have been long used in online shopping, providing customers with a set of recommended items. However, travel recommendation is a more complex issue as tourists take both budget and time limitations into consideration (Herzog & Wörndl, 2014). In addition, things can become even more complicated if the system aims to analyze the trip routes as well and propose the best possible options according to the available money and preferred time frame. This is referred to as the *tourist trip design problem* (Souffriau *et al.*, 2008) which is an extension of the *orienteering problem* where the objective is to maximize the sum of the scores of the location candidates while satisfying all the limitations (Vansteenwegen & Van Oudheusden, 2007).

As discussed earlier, a variety of algorithms exists for designing and implementing the recommendation systems in different applications and areas. Focusing on different aspects of tourism industry, researchers have recently concentrated more on travel recommendation systems. Using online travel information, Liu *et al.* (2011) proposed a personalized travel package recommender. They developed a Tourist-Area-Season Topic model (TAST) which considered tourists' interests along with a number of extracted features such as region, location

[4] GDP can be thought as the size of economy. It represents the total value of all goods and services produced within a specific time period.
[5] World's largest travel site. For more information, see: http://www.tripadvisor.com
[6] http://www.expedia.com

and season of the travel. The TAST model was then used in a cocktail approach to make the final personalized recommendations.

Case-based systems can be regarded as a sub-category of the content-based algorithms where the items are structured as different cases based on their features, and the created cases are compared to a given user's profile and interests to recommend the most similar cases (Smyth, 2007). Such recommender systems have been already applied for travel recommendations. A specific example is when the cases are grouped together in order to recommend a longer trip while still satisfying the user's limitations. However, in most of the implemented recommenders in this area, the dependencies between different parts of the case are not considered. In addition, calculating the score (rating) for all the possible combinations is not always computationally feasible (Herzog & Wörndl, 2014).

Therefore, heuristics are mostly applied here which recommend a local optimum. Xie *et al.* (2010) developed a composite recommender system which can leverage from one or more internal smaller recommenders, focusing on both the quality of recommendations and run time, to suggest the top-*n* packages. In another study, Angel *et al.* (2009) proposed an algorithm which provides the best recommendations, using a set of keywords. The approach is based on bundling the recommendation packages through analyzing the relationships and associations among different entities in the system. They also employed early pruning techniques and termination strategies to increase the efficiency of the algorithm. In another work, Souffriau *et al.* (2008) tackled the tourist trip design problem (TTDP) and analyzed its relation with the orienteering problem and presented an algorithm which is based on the guided local search meta-heuristic approach. Yang *et al.* (2011) also focused on TTDP and designed a cost-aware travel recommender system. In a similar study, Lu and his colleagues (2011) proposed an optimal trip recommender and planning system, named Trip-Mine, which mainly concentrates on travelers' time limitations.

Ricci *et al.* (2006) proposed a trip recommendation methodology, named Trip@dvice, which builds a case-based recommender that supports the selection of the travel products, *e.g.* hotels, and also suggests a travel plan. The systems stores recommendation sessions and uses them as cases. To improve the recommendations, users can give direct feedback and make changes on queries during the recommendation process. Schumacher and Rey (2011) analyzed

recommender systems for dynamic packaging of tourism services and claimed that case-based recommenders and association rules are the best solutions for tourism recommender systems. In a recent study, Chen and his colleagues (2013) build a hybrid recommender system which uses item-based collaborative filtering to predict the interests of a given tourist through collecting information from other tourists. Their proposed system uses genetic algorithm and minimizes a cost function to find the best traveling route out of a set of candidates. Tan *et al.* (2014) proposed an object-oriented recommender system as a framework for developers for designing personalized travel package recommender systems. The additional context information can be also imported to the framework, where different types of the additional information are considered as features. The object is next defined as a set of features and topic modelling approach or Bayesian networks are used to extract the implicit relationships among the objects. In the next section, research gaps are discussed.

2.9 Research Open Problems in Recommender Systems

As discussed, recommender systems have been studied before. Recently, a new wave of interest in travel recommender systems has been observed in computer science community. Although a number of works has been done in this area, the problem of exploiting unique and case-specific features to design highly accurate personalized travel recommender system is still open (Liu *et al.*, 2011). In other words, there exists no general best approach which outperforms all the available algorithms in all situations. The design of the travel recommender system is highly dependent on the available data, the explicit and implicit data features, and the target market and application. Several studies focused on the mentioned drawbacks of recommender systems, mainly CB and CF methods, during the last decade trying to improve the techniques. Although it is widely accepted that recommender systems can be regarded as useful tools for providing users with tailored information and recommendations, due to the broad range of applications more effort is needed to design well-established solutions for real-life problems (Park *et al.*, 2012).

The design and implementation of personalized hotel recommenders face several challenges. The first challenge is the sparsity of customers' travel information in comparison with other products such as books or movies. One reason might be the higher cost of travel in comparison with other entertainments, *e.g.* going to the cinema, which has led to fewer travel

data. In addition, it is quite common to watch several movies in a month or even watch a brilliant movie a number of times, however, it rarely happens that an average customer goes on a trip more than two times per year. Thus, the feedbacks that are used in travel recommender systems are also sparser than other traditional recommenders, calling for a better feature engineering and exploiting both implicit and explicit feedbacks.

As mentioned, a lot of work has been done on improving the algorithms used in recommender systems. However, more research should be done in real-life specific problems/situations, as the field of recommender systems is less mature than other research areas (Park *et al.*, 2012). In addition, using multiple-source data types for extracting implicit and explicit domain-specific features is rarely seen in similar studies. To address the mentioned gaps and weaknesses and to tackle the challenges, a personalized highly intelligent hybrid hotel recommender system is designed and implemented in this thesis. The system uses multiple sources of data for feature extraction. In particular, a sentiment analysis module is implemented which goes through the users' reviews, in form of textual data, and extracts implicit features and scores which are used in the recommendation engine. To the best of our knowledge, this is the first time that an automatic comprehensive sentiment detection module is used inside a hybrid hotel recommender system. A huge Twitter dataset is used to train the sentiment analysis module. In addition, an automatic keyword extraction module is designed to analyze the users' reviews on each hotel and extract complementary implicit data features on hotels for users as well. Moreover, to improve the accuracy and the speed of the recommender system, users are divided into different clusters based on their interests and similarities which are further used to provide the recommendations for a given user. Finally, multiple-criteria rating system is also embedded into the recommender engine to select the best possible hotel candidate(s) for a user. This is also rarely seen in hotel recommender systems where most of the approaches are based on single-rating system. The data collection procedure and design of the proposed system are discussed in detail in Section 4.

3 RESEARCH OBJECTIVES

The general objective of this research is to design and implement a personalized, highly intelligent multi-criteria hybrid hotel recommender system. The system will use a number of machine learning techniques and technologies to propose a tailored recommender system, to be used in a Montreal-based Startup Company. For the first time in the field, a comprehensive sentiment analysis and topic modeling (keyword extraction) will be applied on users' reviews to extract implicit features about users and hotels, to be used in a hybrid recommender engine along with a number of explicit data features of different types. The specific objectives of the research are:

- Extract and collect required data for training the sentiment analysis module to be applied on users' reviews. Data should be labelled reflecting the positivity and negativity of sentences.
- Automatically extract keywords out of users' reviews for each hotel and user, and use them as implicit features in the recommender engine, reflecting implicit interests and highlights for each hotel as well as a user.
- Extract the required travel data from multiple online sources and integrate them into a single dataset. The data will be used for training and building the models.
- Build a multi-criteria data tensor which covers ratings on several aspects of the hotels (and as a result the travel package) such as the overall rating, location rating, value rating, cleanness rating, *etc*. The multi-criteria recommender model will be built on the data tensor.
- Design and implement highly accurate well-tailored hybrid recommender system, able to satisfy the users' needs as well as the business requirements.
- Test the performance of the proposed approach and validate it against real-life business data through running various experiments.

4 DATA AND METHODOLOGY

In this section, the data and the methodology of the research are presented and discussed. The data collection procedure will be discussed in details and the detailed design of the recommender system is presented.

4.1 Data

The data required for this research can be divided into two separate parts. One comprehensive dataset is required for training the sentiment analysis module. Twitter[7] was selected as the data source for this purpose. Another dataset is needed for training the recommender system as well as performing automatic keyword extraction. This dataset should contain comprehensive information about hotels, users, and users' reviews. In this section, the data extraction and integration is discussed in details.

4.1.1 Twitter Dataset

Machine learning methods typically require large amounts of data to provide sufficient predictive power. Since no labelled large-scale travel-specific training corpuses exist to this date, we decided to use other *user expression* resources. We chose *Twitter* as the source of the training data. Twitter is a social networking service in which users can post real time messages, called *tweets*. In particular, a corpus of 1.6 million tweets[8], labelled as positive and negative, was collected from the Internet and used as the training dataset.

4.1.1.1 Preprocessing of Twitter Dataset

Since people mainly use quick and short messages and due to the automatic collection of the data, the data was not clean hence we first preprocessed and cleaned the collected Tweets dataset. For this purpose, we removed the hashtags[9] and hyperlinks from the tweets. In addition, we removed the target users[10] from the twitter data and trimmed the texts. The final dataset is referred to as *Tweets Database* in the rest of thesis. The whole process is depicted in Figure 1.

[7] http://www.twitter.com
[8] The original dataset was taken from (Go, Bhayani, & Huang, 2009).
[9] Users of Twitter usually use hashtags to refer to or mark topics.
[10] Users of Twitter use the "@" symbol to refer to other users.

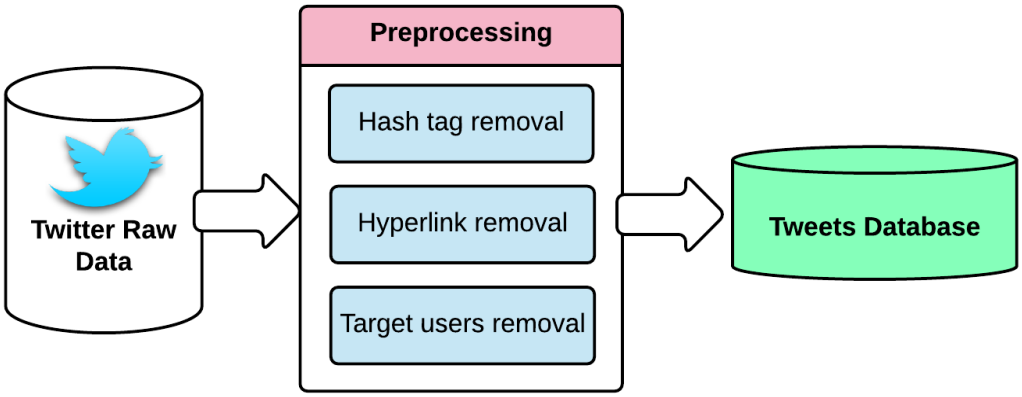

Figure 1. Twitter dataset, preprocessing procedure. The raw Twitter dataset, containing a corpus of 1.6 million tweets, is preprocessed through several modules. The hash tags and hyperlinks are removed from the tweets. And, the target users are eliminated. The preprocessed clean data are stored in *Tweets Database*.

4.1.2 Travel Dataset

TripAdvisor[11] was mainly used as the data source for hotel and travel packages in this research. TripAdvisor employs user-generated content, and its data have been used in a number of studies in the field of travel recommender systems. The TripAdvisor website is free to use and the company's business plan is based on the support from advertisement. Hence, the availability of the data as well as the possibility of comparing the results with similar studies that used TripAdvisor as the input data source were some of the main reasons for selecting TripAdvisor.

We collected the TripAdvisor travel data from multiple sources since we wanted to have a complete corpus of user reviews, hotel ratings (multi-aspect), as well as complete hotel information. For this purpose, a complete list of hotels including all the respective information about the hotels such as class, region, physical address, website, *etc.* as well as users' reviews on the listed hotels were downloaded from Mr. Jiwei Li's website[12]. This comprehensive dataset, which was used in a number of research papers, contained 878,561 users' reviews on 4,333 different hotels, about 1.3 Gigabytes, which was crawled from the TripAdvisor website. The original data was in JSON[13] format. In addition, we used another dataset which contains 246,400 hotel reviews. The data was scraped from TripAdvisor by Mr. Hongning Wang[14] within a one

[11] An American company which is the world's largest travel website. For more information, see: http://www.tripadvisor.com
[12] Ph.D. student at Stanford University, http://www.cs.cmu.edu/~jiweil/html/ACL_profile_data.html
[13] An open standard format which contains data objects in form of feature-value pairs.
[14] http://sifaka.cs.uiuc.edu/~wang296/

month period, spanning from February to March 2009, and is free to use. Data contains numerical ratings, ranging from 1 to 5, provided by users on different aspects of the hotels, *e.g.* value, room quality, location, and service, along with other complementary information about the hotels. In addition, textual users' reviews on hotels are also available in the file. The data was preprocessed by removing all the excess spaces and tabs and converting commas to semi-colons.

Having collected the required hotel data, we next integrated the collected data into a MySQL[15] database. For this purpose, different entities, *e.g.* hotels and users, were first identified and their unique IDs were considered as the primary key in respective tables. Next, an automatic data integration procedure was coded in JAVA which went through all the records and integrated them into the database, through checking for duplicates and inserting all the related information about an entity in the respective row. The final database contains 4,333 distinct hotels with complete information about them. 148,429 users have rated 1,850 different hotels focusing on various aspects. In addition, 148,421 users have written text reviews about the hotels in the integrated dataset. The integrated database is referred to as the *Integrated Travel Data* in the rest of the thesis. The whole travel data collection and integration procedure is depicted in Figure 2.

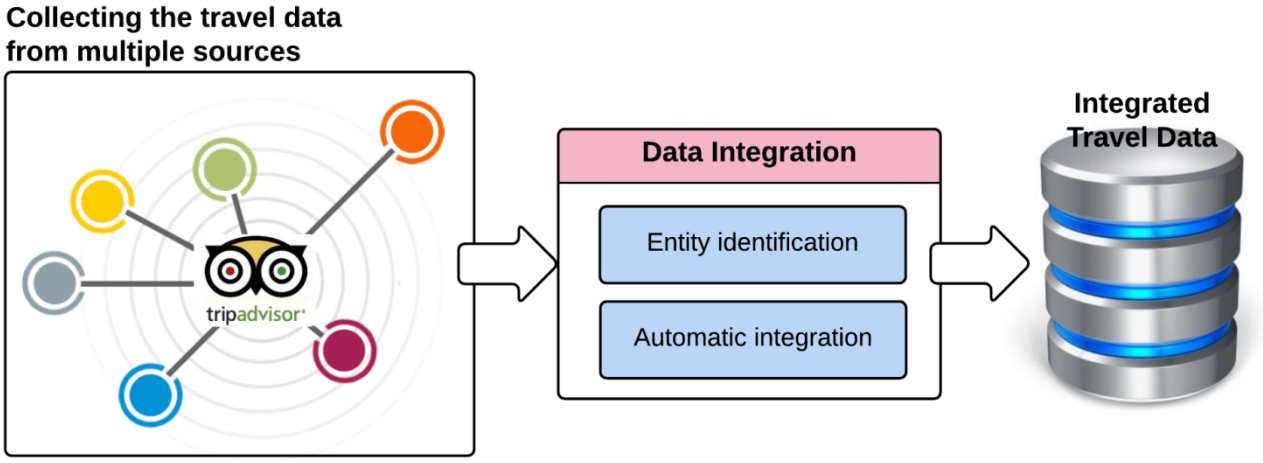

Figure 2. The travel data collection and integration procedure. TripAdvisor was selected as the target data source. The required data were collected from multiple sources, each covering different aspects of TripAdvisor. The collected data was then automatically integrated into a single MySQL database, named *Integrated Travel Data*.

4.1.2.1 Integrated Travel Data, Preprocessing

[15] An open-source relational database management system. For more information, see: http://www.mysql.com

The Integrated Travel Data needed cleaning mainly because it was collected from multiple sources. Therefore, several data processing tasks were performed on the mentioned database, including noise removal, *i.e.* removing the rows with so many missing values, hyperlink removal, performing spell check on the collected data, and converting all the words into lower cases. Further preprocessing, *e.g.* stop words and punctuations removal, will be performed later on, if required in the task. The clean final travel database is referred to as the *Target Data* in the rest of the thesis. The overall preprocessing procedure is depicted in Figure 3.

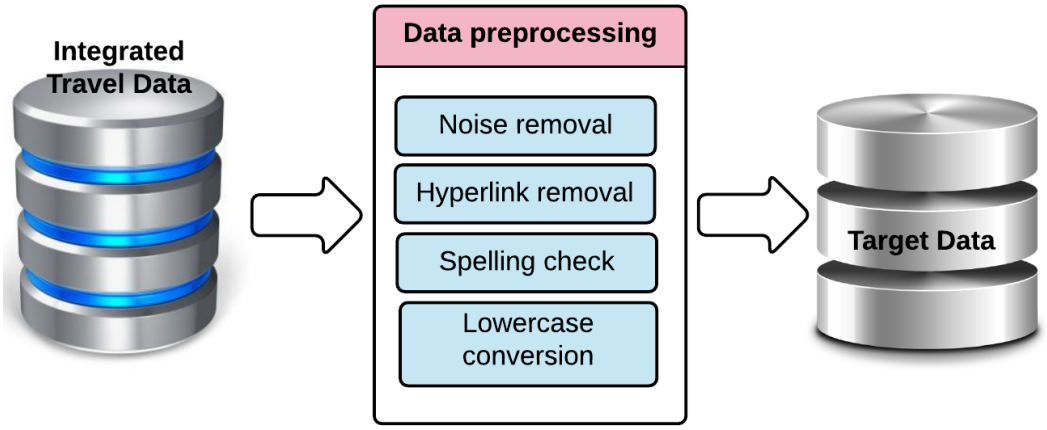

Figure 3. Integrated Travel Data, preprocessing procedure. Further cleaning and preprocessing were performed on the Integrated Travel Data through removing noisy data, checking for the spelling errors, eliminating the hyperlinks, and converting the data to the lowercase. The processed travel data were stored in another dataset named, the *Target Data*.

4.2 Methodology

In general, this research employs a number of different tools, methods and methodologies for designing and implementing the hybrid hotel recommender system (Ebadi & Krzyzak, 2016b). Machine learning[16] techniques and natural language processing (NLP)[17] were employed to develop the core of the recommender engine. SQL[18] queries were used for data collection and manipulation. The entire system was coded in Python[19] programming language. The system contains three different sub-systems, *i.e.* sentiment analysis module, keyword extraction module,

[16] A sub-field of computer science which mainly deals with developing computer systems which can learn to act without being explicitly programmed.
[17] A sub-field of computer science which is mainly about the human-computer interaction and human language processing.
[18] Structured Query Language (SQL) is a computer language for data management and manipulation.
[19] A general-purpose high-level programming language.

and the recommender engine, which are separately presented and discussed in details in this section.

4.2.1 Sentiment Analysis Module

Users' reviews are very informative and can significantly help the recommender engine. To be able to assess users' opinions more accurately, an intelligent sub-system was designed to extract implicit features from users' reviews. This sub-system automatically detects the positivity and negativity of users' text reviews[20] and provides the recommender engine with a complementary implicit feature, in form of a polarity score, reflecting user's satisfaction. This procedure is known as *polarity detection* or *sentiment analysis* in computer science literature, where various techniques such as machine learning techniques, natural language processing (NLP) and computational linguistics methods are employed to extract, identify or characterize subjective information/impression (*e.g.* polarity) from the textual input data (Turney, 2002).

The highly subjective nature of humans makes automatic polarity detection a very challenging task. Opinions are mostly expressed in complex ways, using rhetorical modes such as sarcasm, which makes the automatic identification difficult. Manual labelling of the large-scale data if not impossible, is a very laborious task. To solve this problem, machine learning techniques can be used to provide an automated solution to this problem. We applied machine learning and text processing techniques to provide such an automated mechanism. The overall architecture of the intelligent sentiment analysis sub-system is shown in Figure 4.

[20] It can be also applied on any other type of the text input.

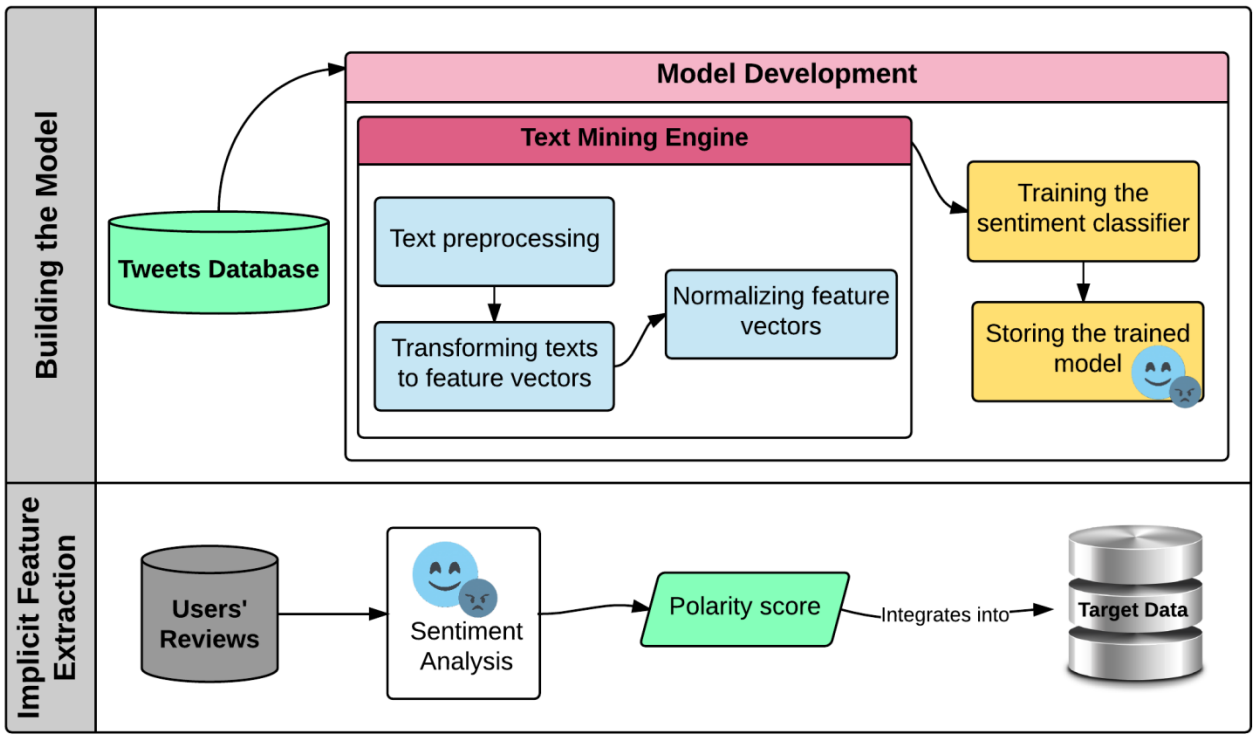

Figure 4. The intelligent sentiment analysis sub-system. The model is first trained and built using the Tweets Database. A text mining engine was designed and implemented to preprocess the data, and to transform the textual data to feature vectors that are appropriate for the machine learning classifier. The transformed data were used to train the machine learning classifier, and to build the sentiment analysis model. The trained model was stored and then used to detect the polarity of users' reviews and to extract the implicit features. For this purpose, a polarity score is calculated, reflecting the polarity of a given review, along with its intensity.

As explained before, the Tweets Database was used to train and develop the sentiment analysis model. For this purpose, a text mining engine was designed which takes the Tweets as the input and converts them to numerical form, suitable for training the sentiment classifier. The input text was already converted to lower case at the time of integrating the travel data. The punctuations were also replaced with a blank space. Next, the streams of text were broken into the smallest meaningful components called *tokens*. We then converted the tokens to *stems* by removing and replacing the word suffixes to obtain the common root of the word. We will refer to tokens and stems as *tokens* in the rest of the thesis.

To improve the accuracy of the algorithm, we considered both unigrams and bigrams[21] of the tokens. In order to obtain numerical feature vectors, all the considered tokens were then converted to count vectors in which the index value of a token was linked to its frequency in the

[21] In general, an *n-gram* is a contiguous sequence of *n* items (tokens or stems) from a given sequence of text.

whole training corpus. To normalize the feature vector, we employed the term frequency-inverse document frequency (*tf-idf*) approach. Tf-idf takes the importance of a word (token) to a document in a corpus into consideration and is often used as a weighting factor. In other words, tf-idf helps to adjust the feature vectors by considering that, in general, some words (tokens) may appear more frequently. Theoretically, tf-idf is the product of two statistics, namely term frequency and the inverse document frequency (Brusilovsky, Kobsa, & Nejdl, 2007). The term frequency (tf) is calculated by counting the number of times that a term *t* occurs in document *d*. The inverse document frequency (idf) focuses on the amount of information that each term provides and is defined as in Equation (1).

$$idf(t, D) = log \frac{|D|}{|\{d \in D: t \in d\}|} . \qquad (1)$$

In Equation (1), $|D|$ is the total number of documents in the dataset, and $|\{d \in D: t \in d\}|$ is number of documents in which the term *t* appears. Normalizing the feature vector was the final step in preparing the data for training the model. The entire process is depicted in Figure 4 as the *text mining engine*.

After preprocessing the text data, a logistic regression model is built using a 10-fold cross validation approach to assure the accuracy of the learned model. In 10-fold cross validation approach, the data is split into 10 equal sized sets named as *folds*. One of the folds is set aside for validation while the other 9-folds are used for training and fitting the model. This process is repeated 10 times by setting aside a different fold of the 10-folds and the accuracy metrics are calculated by testing the model on the aside fold. This gives a total of 10 different runs of fitting and checking the logistic regression model. Cross validation is used for model evaluation. One of the advantages of cross validation is that it finally considers all the examples in the dataset as training and test sets, meaning that after cross validation all the data points have been considered once for training and once as the test set. Thus, it lowers the effect of dividing data into separate training and test sets, however, the disadvantage of this evaluation method is that the training procedure should be started *k* times (number of folds) from the scratch, which makes the procedure time consuming. The sentiment analysis model is stored as the output of this phase. The whole phase is marked as *Building the Model* in Figure 4, and the sentiment classifier design is shown in Figure 5. The built model is able to detect polarity at both *sentence* and *word* levels.

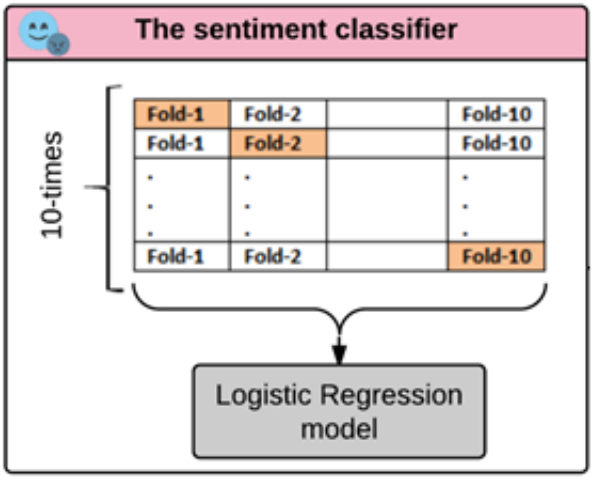

Figure 5. The 10-fold cross validation design of the sentiment analysis module. The data is split into 10 equal size sets, named as folds. The process is run 10 times. In each run, one set is fold aside for validation and the other 9 folds are used for training the model. The accuracy of the model is calculated on the test fold.

The model was found to be 85% accurate in predicting polarity of Tweets. Thus, the model shows promising performance in predicting the sentiment of a given sentence/text comparing to the literature. For example, Pak and Paroubek (2010) achieved 81% of accuracy in predicting positivity and negativity of tweets. In a recent survey study, Rosenthal *et al.* (2015) listed the performance results of 11 different systems in predicting phrase-level binary polarity of tweets, where all the systems have accuracy lower than 85%. Using the stored model, the polarity of the given user's review is predicted and an intermediate output, namely the *polarity score* is generated, as shown in Figure 4. The system reports a value in the range of [-1, 1] as the polarity score where the score reflects the intensity of polarity: more negative values represent more negative sentiment, and more positive values indicate more positive sentiment. That means, from the polarity score the overall sentiment of the given user's review as well as its intensity is detected. The polarity score and the detected sentiment of the review are stored in the Target Data as implicit features which will be used as complementary ratings data in the recommender engine.

4.2.2 Keyword Extraction Module

Summarizing and analyzing the content of the users' reviews can be very beneficial. The main objective of the keyword extraction sub-system is to go through the users' reviews and extract keywords out of them, and assign them as implicit features to the respective hotels and

users. These features can be regarded as points of interests about different hotels that can be further used in the hotel recommender engine. Keyword extraction is mainly used in the text mining context (*e.g.* Rajman & Besançon, 1998)[22], and has a wide range of applications, *e.g.* in document retrieval and document clustering.

In general, machine learning techniques that are employed for classifying a given corpus are called clustering. Clustering that is widely in use in various scientific fields and applications (*e.g.* bioinformatics, image segmentation, and document summarization) is an unsupervised learning technique that discovers the hidden groupings in a dataset. Topic modeling is a clustering technique in which a collection of documents are automatically organized into a set of clusters (topics). Latent Dirichlet Allocation (LDA) is a topic modeling method.

Clustering large quantities of text has several unique challenges comparing to non-text data mining tasks. The two main concerns are the highly unstructured nature of the text data that requires encoding to a form that is recognized by clustering techniques, and the high dimensionality of the encoded data (Millar, Peterson, & Mendenhall, 2009). Despite the challenges, topic modeling has been widely used in several studies for automatic extraction of the semantic or thematic topics from large corpus of documents (*e.g.* Griffiths & Steyvers, 2004; Griffiths, Steyvers, & Tenenbaum, 2007; Weng *et al.*, 2010). In addition, topic modeling can be easily generalized to treat other data types, *e.g.* image analysis (Fei-Fei & Perona, 2005; Sivic *et al.*, 2005), survey data (Erosheva, 2002), and biological data (Pritchard, Stephens, & Donnelly, 2000).

In this thesis, the keyword extraction sub-system runs on all the users' reviews on different hotels, and employs the LDA method to extract the topics out of the set of reviews. The extracted topics are then refined to find the representative set of keywords for each user and hotel in the Target Data. This can act as a set of interests for each specific user or a set of keywords for each hotel. The refinement stage was added to the sub-system as LDA might result in soft clusters where a semi-automatic refinement procedure can filter the undesired results and improve the accuracy. The whole keyword extraction procedure is depicted in Figure 6. As seen, users' reviews are first collected from the Target Data and are preprocessed. In particular, we

[22] In other contexts, similar research topics are called differently. For example, it is called as *automatic term recognition* in computational linguistics.

removed special characters from the data as they could affect the accuracy of the system negatively. In addition, reviews that were in a language other than English (*e.g.* Chinese, French) were removed. Next, we removed the English stop words[23] from the vocabulary and numerical values from the data as they were not informative in the defined keyword extraction task. The clean users' reviews dataset is then split into a set of reviews for each user. Next, LDA is performed on each set of reviews separately, and the set of specific keywords for each user are extracted. Finally, a semi-automatic procedure refines the extracted keywords. The final extracted keywords are assigned to the users as implicit features which partially reflect their interests. In addition, the extracted keywords are integrated in order to reflect the hotel characteristics implicitly.

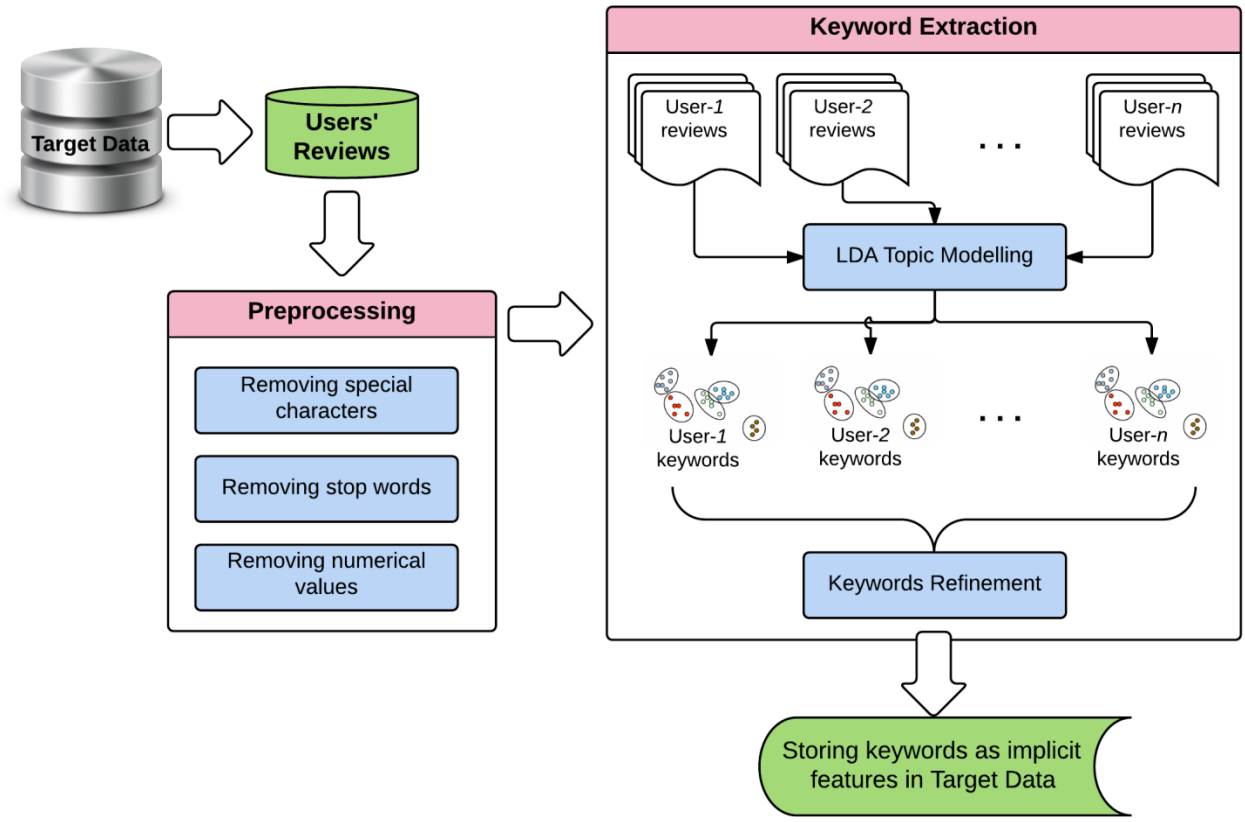

Figure 6. The keyword extraction sub-system. Users' reviews are first extracted from the Target Data and are then preprocessed. The special characters and stop words are removed from the data, and numerical values are also eliminated. The preprocessed data are then fed into the keyword extraction module where machine learning LDA topic modelling technique is used for extracting the keywords for each user based

[23] Stop words are the most common words which are in the text but can be of little value in detecting the most informative keywords.

on their reviews. The detected keywords list is then manually refined. The final result is stored in the Target Data.

As mentioned, in this thesis LDA technique is used for topic modelling and keyword extraction, which was first introduced by Blei, Ng, and Jordan (2003). In this approach, documents (in thesis, users' reviews) are represented as a mixture of topics and topics are modeled according to the distribution of words. Each document is assigned a number of topics and their probabilities. LDA is a generative model[24] that uses a joint probability distribution for both observed and hidden random variables. According to Blei, Ng, and Jordan (2003), the joint distribution of observed and hidden variables is defined as in Equation (2).

$$P(\beta, \theta, z, w) = \prod_{i=1}^{K} P(\beta_i) \prod_{d=1}^{D} P(\theta_d) \left(\prod_{n=1}^{N} P\left(z_{d,n} | \theta_d\right) P(w_{d,n} | z_{d,n}, \beta) \right). \qquad (2)$$

In Equation (2), K is number of topics, D is number of documents, N is the length of a document, β_k is the words distribution in topic k, θ_d is the topics distribution in document d, z_d is the topic assignment for document d and similarly $z_{d,,n}$ is the topic assignment for the n^{th} word in document d, w_d are the words observed in document d and $w_{d,n}$ is the n^{th} word in document d respectively. Based on the definition of the algorithm, a Python script was coded, implementing the LDA method, and was used on the users' reviews data. The number of topics/keywords was dynamically set in the program for each user review of a hotel with a limit to be equal or less than 10, as the maximum number of topics/keywords. Table 1 shows a list of the detected keywords for 10 randomly selected users. In the next section, the design of the recommender engine is presented and discussed.

Table 1. List of detected keywords for 10 randomly selected users

No	Username	Keywords
1	385	Room, Location, Staff, Clean, View
2	007Cambridge	Beach, Resort, Pool, Food, Staff, Time
3	007Danielle	Service, Room, Pool, Staff
4	42289	Room, Bathroom, Bed, Shower, Comfortable,
5	08NOVA	Trip, Time, Read, Book, People, Place, Staff
6	0ntheroadagain	Station, Location, Room, Tokyo, Staff, Clean, Train
7	0u812ic	Airport, Arrive, Staff, Time, Trip, Room, Check, Luggage,
8	1000Islands	Florence, Staff, Breakfast, Helpful, Location, Wonderful, Room

[24] It includes some latent (hidden) parameters. Given the observed variables, distribution of the hidden variables is calculated.

9	101runner	Parking, Room, Night, Car, Street, Day, Valet
10	13Blackwood	Walk, Location, Clean, Station, Amsterdam, Staff, City

4.2.3 The Recommender Engine

Having extracted the keywords out of users' reviews, detecting the sentiments of the reviews, and performing several preprocessing tasks, the data became ready for the recommender engine. The recommender engine itself consists of three major modules: 1) *User Clustering Module*, 2) *Matrix Factorization Module*, and 3) *Hybrid Recommender System*, as shown in Figure 7.

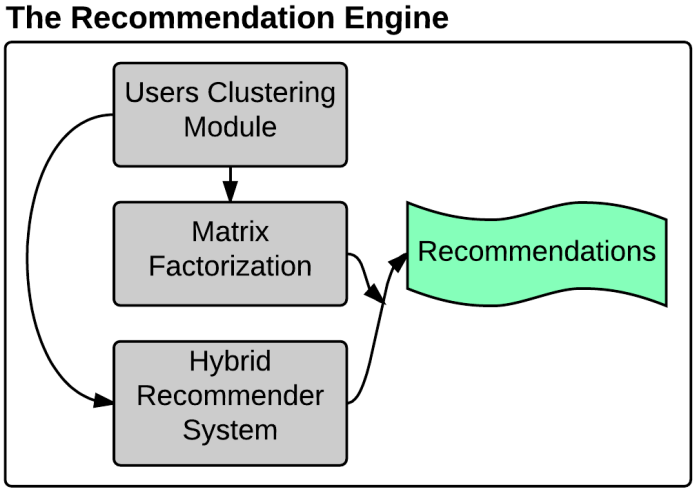

Figure 7. The top-level design of the recommender engine. The system contains three main modules, namely user clustering, matrix factorization, and the hybrid recommender system. Users are first clustered based on various features. The selected cluster is then fed into the matrix factorization module and the hybrid recommender system. The output of the mentioned two modules forms the final recommendations of the system[25].

The *Users Clustering Module* clusters all the users in the system into 4 groups[26]. The distance of any given user's features set is then compared with the centroids of the generated clusters, and the best cluster is selected for the user. Next, the selected cluster is fed into the *Matrix Factorization* and *Hybrid Recommender System* modules, separately, where the final recommendation is made based on the outputs of the two mentioned modules. In the rest of this section, all the modules are presented and discussed in details.

[25] The hybrid recommender system, itself, produces 3 separate recommendation sets that will be explained later in section 4.2.3.3.
[26] The method details, including why 4 groups, are discussed later in this section.

4.2.3.1 Users Clustering Module

A clustering module was implemented and included, mainly due to the high sparsity of the users' data features. Clustering techniques can reduce the sparsity and improve the performance and scalability of the recommender systems (Bilge & Polat, 2013; Nilashi, bin Ibrahim, & Ithnin, 2014). The aim of this module is to categorize users based on their similarities. Thus, before running the recommendation engine, the best cluster of users is selected for the given user. This improves the accuracy of the recommender system as well as the quality of recommendations.

Figure 8 shows the clustering procedure. As seen, two sources of users' data are used in the *Users Clustering Module*, namely users' demographics and the detected keywords. Users' demographics contain information such as user's age, gender, date of registration in the system, location, *etc*. This data are highly sparse as they are optional and not all the users insert such data while registering in a travel advisor website. Therefore, to increase the data dimension, we also included the detected user-specific keywords, obtained from the *Keyword Extraction Module*, as explained before. The *Users Clustering Module* uses the huge user-feature matrix to cluster the users into different groups.

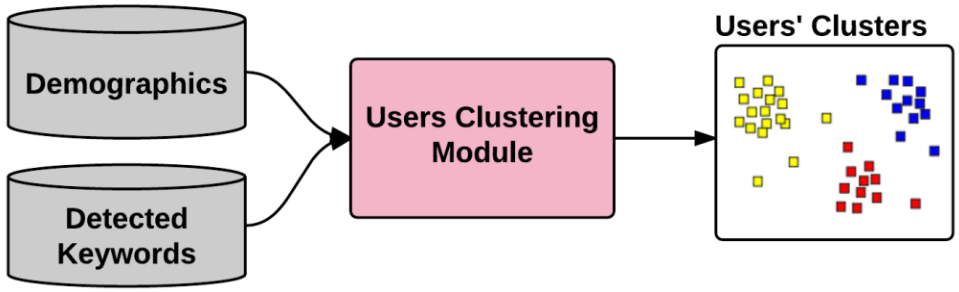

Figure 8. Clustering the users. Two main sources of users' data are applied for the clustering, i.e. users' demographics and the detected keywords. Employing these explicit and implicit features, the users are then clustered using a K-Means based clustering approach.

To perform the clustering, we decided to use *K-Means* clustering approach (MacQueen, 1967). The ease of implementation, speed, and the fact that *K-Means* performs reasonably well in large datasets (Huang, 1997), were the main reasons for such selection. However, since the data contained both categorical and numerical variables (features), we used a variation of *K-Means*, named *K-Prototypes* (Huang, 1997), which is able to handle both categorical and numerical

features. Figure 9 shows the internal design of the *Users Clustering Module*, implementing the *K-Prototypes* algorithm.

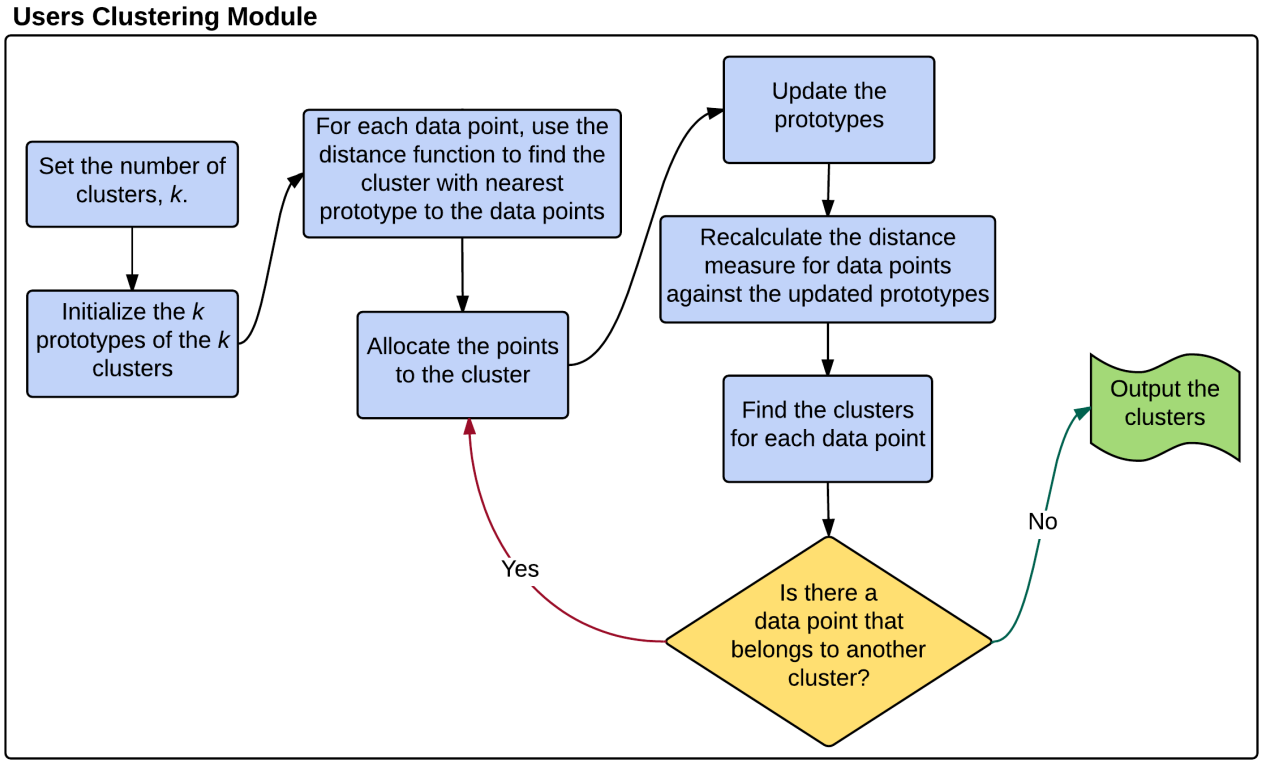

Figure 9. Internal design of the *Users Clustering Module*. The module implements *K-Prototypes* clustering algorithm. The algorithm is similar to *K-Means* but it works with both numerical and categorical features. *K* prototypes are first randomly initiated, and using a distance function, data points are placed in the nearest prototype. The prototypes are then updated, and the procedure is repeated until no further movement of data points between the clusters is possible.

In *K-Prototypes* algorithm, data points are clustered against k prototypes. *K-Prototypes* uses decision tree induction algorithms to generate rules for the clusters, which helps to increase the interpretability of the algorithm and to detect the clusters of interest more accurately (Huang, 1997). Suppose, we have a set of n data points, $D = \{D_1, D_2, \dots, D_n\}$, where each data point has m features, let us say $D_i = [d_{i1}, d_{i2}, \dots, d_{im}]$. The goal is to partition D into k disjoint clusters such that the inter-cluster distance of the data points is minimized, while maximizing the intra-clusters distance. The distance measure is thus defined in *K-Prototypes* algorithm (Huang, 1997) as in Equation (3).

$$E = \sum_{i=1}^{k} (E_i^r + E_i^c) = \sum_{i=1}^{k} E_i^r + \sum_{i=1}^{k} E_i^c = E^r + E^c. \qquad (3)$$

In Equation (3), E is the total distance function for clustering n data points with numerical and categorical features. E^r is the sum of the distances for the numerical features over all the k clusters and E^c is the sum for the categorical attributes. Both E^r and E^c are non-negative, thus E is minimized through minimizing E^r and E^c. The squared Euclidean distance is used for calculating the distance function for numerical values. And, the distance function for categorical features is based on the number of mismatches between the data points and the cluster prototypes.

As seen in Figure 9, the number of clusters/prototypes (k) is the parameter that should be provided to the algorithm. As discussed earlier, clustering goal is to group items, here users, in sets/clusters such that the items within a cluster have the highest similarity, whereas the similarity is minimum for the items from different clusters. Thus, finding the optimal number of clusters, the best k, is crucial. We used the *Gap statistic* (Tibshirani, Walther, & Hastie, 2001) for estimating the best k for the users data. The Gap statistic technique can be applied in any clustering approach, *e.g.* K-Means or K-Prototypes. It considers an appropriate reference null distribution and compares the change in within-cluster dispersion. Theoretically, the Gap statistic approach is a way to standardize the comparison of $\log W_k$ with a null reference distribution of data with no clear clustering, where W_k is the within-cluster dispersion. Suppose, x_i and x_j are two given data points in a given cluster, named C_k, which contains n_k data points. Then, the sum of intra-cluster distances between the points in C_k is defined as in Equation (4).

$$D_k = \sum_{x_i \in C_k} \sum_{x_j \in C_k} || x_i - x_j ||^2 = 2n_k \sum_{x_i \in C_k} || x_i - \mu_k ||^2 . \qquad (4)$$

In Equation (4), μ_k is the centroid of C_k. Through summing up the normalized values of D_k over the K clusters, as stated in Equation (5), W_k is obtained which can be regarded as a measure of the compactness of the clustering approach.

$$W_k = \sum_{k=1}^{K} \frac{1}{2n_k} D_k. \qquad (5)$$

Now, the estimated optimal number of clusters, *i.e. K*, is detected by the Gap statistic, where *K* is optimal if $\log W_k$ places the farthest below the curve of the reference distribution. Equation (6) defines the Gap statistic.

$$Gap_n(k) = E_n^*\{\log W_k\} - \log W_k. \qquad (6)$$

In Equation (6), E_n^* is the expectation under a sample size *n* from the reference distribution. We generated the reference dataset by sampling uniformly from the original users' data. Next, 10 different replicates were generated through *Monte Carlo* sampling from the reference distribution, and the average of $\log W_k$ was considered as the estimation of $E_n^*\{\log W_k\}$. Finally, S_k is defined based on the standard deviation of the obtained $\log W_k$ from 10 Monte Carlo replicates, *i.e.* SD_k. This measure accounts for the simulation error, and is defined as in Equation (7).

$$S_k = \sqrt{1 + \frac{1}{10}}\, SD_k. \qquad (7)$$

The optimal number of clusters (*K*) is the smallest *k* for which Equation (8) holds (Tibshirani, Walther, & Hastie, 2001).

$$Gap(k) \geq Gap(k+1) - S_{k+1}. \qquad (8)$$

The Gap statistic approach was designed, implemented, and applied on the users-features matrix, explained before. Figure 10 shows the Gap statistic graph versus various numbers of clusters. As seen, the Gap statistic peaks at k = 4 with the value of ~1.006. We further investigated the issue by plotting $DG_k = Gap(k) - (Gap(k+1) - S_{k+1})$. The results are seen in Figure 11. In Figure 11, the optimal *k* is the smallest *k* for which DG_k becomes positive. As it is observed, the existence of 4 clusters is confirmed. Thus, according to Figure 10 and Figure 11, the number of clusters was set to 4 in the *Users Clustering Module*.

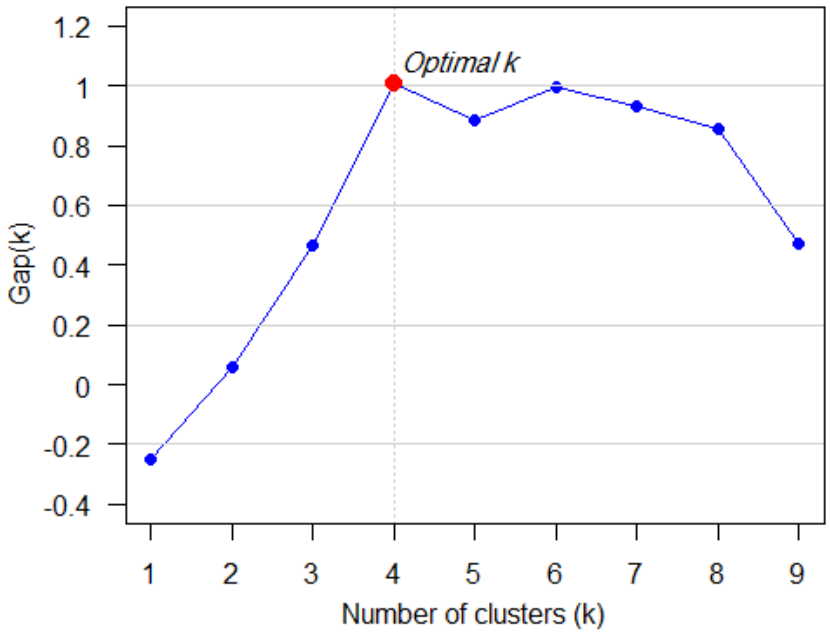

Figure 10. Gap statistic vs. various number of clusters. The statistic is maximized at $k = 4$, indicating the estimated optimal number of clusters.

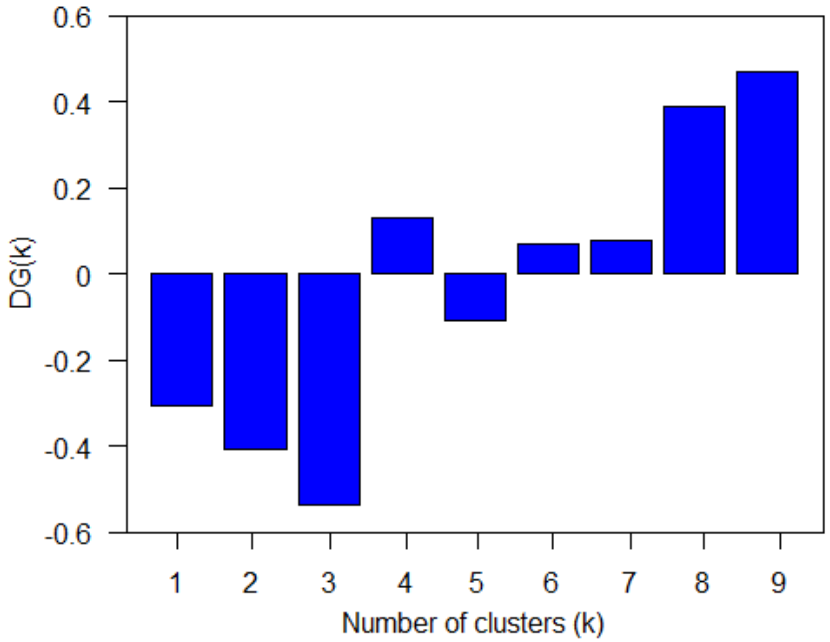

Figure 11. The trend of $\boldsymbol{DG_k = Gap(k) - (Gap(k+1) - S_{k+1})}$ vs. various numbers of clusters. $\boldsymbol{DG_k}$ becomes positive for the first time at $k = 4$, which confirms that the data contains 4 different clusters.

4.2.3.2 Matrix Factorization Module

In recommender systems, the *cold start* is a well-known problem. This refers to the fact that the recommender system is not able to provide any recommendation for users or items that has not enough information about. This might cause the system to make common and/or the same recommendations to the users. Specifically in the case of collaborative filtering, the system works by identifying the users with similar/same preferences to the given user, and then recommends the items that those users, but not the given user, have already favored. Thus, this will fail to suggest items for which there exist no ratings, *e.g.* new items to the community (Schein *et al.*, 2002). One solution is to apply matrix factorization techniques.

Matrix factorization techniques enable the system to infer latent features that are hidden in the inter-actions between users and items. It is also argued in the literature that matrix factorization methods can enhance the accuracy of collaborative filtering recommendations, improve the scalability, and provide more flexible solutions to real-life problems (Koren, Bell, & Volinsky, 2009). In the case of travel recommendation system, in general, there exists a set of users, and a set of hotels, where each user has rated some hotels. Table 2 shows a sample dataset of users, hotels, and ratings. As seen, there is some blank cells in the table, indicating the hotels that a specific user has not already rated (has not already been there). Matrix factorization will help to predict the missing ratings such that the predicted ratings are consistent with the existing values in the main matrix.

Table 2. A sample dataset of user-hotel ratings

	Hotel-1	**Hotel-2**	**Hotel-3**
User-1	4	3	-
User-2	5	-	2
User-3	-	-	4

Matrix factorization can be regarded as a mathematical tool for manipulating matrices. In matrix factorization, users and items are characterized by vectors based on the hidden rating pattern in the data, where the main matrix can be then obtained by dot production of the derived vectors. In particular, non-negative matrix factorization (NMF) approach was used in this thesis (Sra & Dhillon, 2005). NMF has been widely used in various fields and applications, including recommender systems (Gemulla *et al.*, 2011; Bao, Fang, & Zhang, 2014). In this approach, the given matrix M, user-item ratings in this case, is (usually) factorized into two matrices, namely P and Q, such that all the three matrices include only non-negative elements. Thus, it is an

approximation of the given matrix by the two derived matrices. Mathematically, let us suppose we have a set of users, *U*, and a set of hotels, named as *H*. Now, consider *M* of size $|U| \times |H|$ as the matrix that contains users' ratings on different hotels. Thus, the task is to find *P* and *Q* matrices such that, $M \approx P \times Q^T$. Therefore, each row in *P* will indicate intensity of the relation between a user and the features. In a similar manner, each row of *Q* will demonstrate the strength of the relationship between a hotel and the features. Suppose, we have *K* latent features hidden in the data.

To obtain a prediction of how user u_i will rate hotel h_j, we need to calculate the dot product of the two derived vectors which are corresponding to u_i and h_j, as stated in Equation (9).

$$m'_{ij} = p_i^T q_j = \sum_{k=1}^{K} p_{ik}\, q_{kj}. \quad (9)$$

To find *P* and *Q*, we used the *gradient descent* approach. We initialized the mentioned matrices with random values, checked how closely they are approximating *M*, and tried to improve the derived matrices through minimizing the difference with *M*, iteratively. For calculating the difference, we considered the error between the estimated rating and the exact one, for each user-hotel pair, as stated in Equation (10). Regularization was also used to avoid over-fitting.

$$e_{ij}^2 = \left(m_{ij} - m'_{ij}\right)^2 + \frac{\beta}{2}\sum_{k=1}^{K}(\,||P||^2 + ||Q||^2). \quad (10)$$

As seen in Equation (10), the squared error was considered in order to obtain a positive error estimate. The next step was minimizing the error estimate. For this purpose, the gradient at the current values should be calculated. Therefore, Equation (10) was differentiated with respect to the two variables. This resulted in update rules stated below as in Equation (11). In Equation (11), α is the learning rate constant, *i.e.* the rate of approaching the minimum[27].

$$p'_{ik} = p_{ik} + \alpha \frac{\partial}{\partial p_{ik}} e_{ij}^2 = p_{ik} + \alpha(2e_{ij}q_{kj} - \beta p_{ik})$$

[27] α was set to a very small value, i.e. 0.0001, in order to prevent the risk of oscillations around the local minimum.

$$q'_{kj} = q_{kj} + \alpha \frac{\partial}{\partial q_{kj}} e_{ij}^2 = q_{kj} + \alpha(2e_{ij}p_{ik} - \beta q_{kj}). \qquad (11)$$

Using the update rules and the mentioned equations, the matrix factorization module was implemented. The overall design of the matrix factorization module is depicted in Figure 12. The input to the module is the selected cluster of data for the given user.

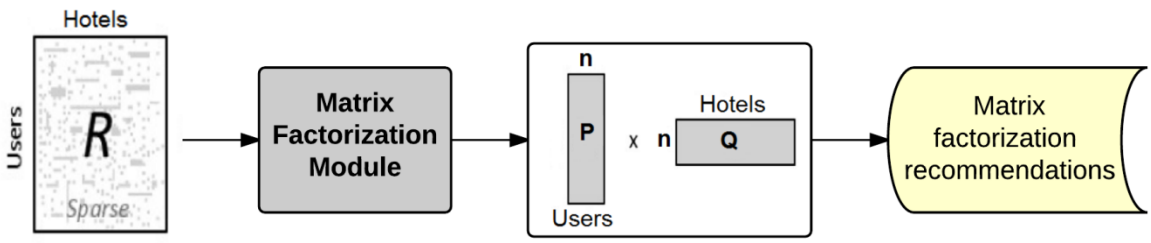

Figure 12. The matrix factorization module. The module takes the selected user cluster as the input and decomposes the users-hotels matrix into a product of two matrices.

4.2.3.3 Hybrid Recommender System

The main goal of this module is to leverage from all types of features to build a hybrid model which is able to recommend hotels to the users more accurately. For this purpose, a set of features which reflects both users and hotels characteristics (separately) are used. These features are called as explicit features. In addition, as discussed earlier, a complementary set of implicit features, *e.g.* users' reviews polarity, is also employed in the model. That is, we searched for the content in the users' reviews that was frequently associated with the hotels in the data. The sentiment analysis module, and the tf-idf approach in particular, along with the keyword extraction module, enable the recommender engine to exploit the content as well, thus, providing more dimension to the system.

As discussed earlier (Figure 7), first the best cluster is identified for a user based on various features and characteristics, and then the hybrid recommender system is provided with the selected user cluster. The overall design of the hybrid recommender system is shown in Figure 13. The hybrid recommender system consists of three sub-modules: 1) User-based collaborative filtering module, 2) Item-based collaborative filtering module, and 3) Multi-criteria recommender system. The hybrid system generates four separate outputs, namely the recommendations generated by the user-based and item-based collaborative filtering modules, as well as the multi-criteria recommender system, along with an integrated composite set of

recommendations. The integrated recommendation is formed by combining the outputs of the three sub-modules and forming a composite set of recommendations. The recommendations can be made for any given user. The mentioned recommender sub-modules are discussed in more detail in the rest of this section.

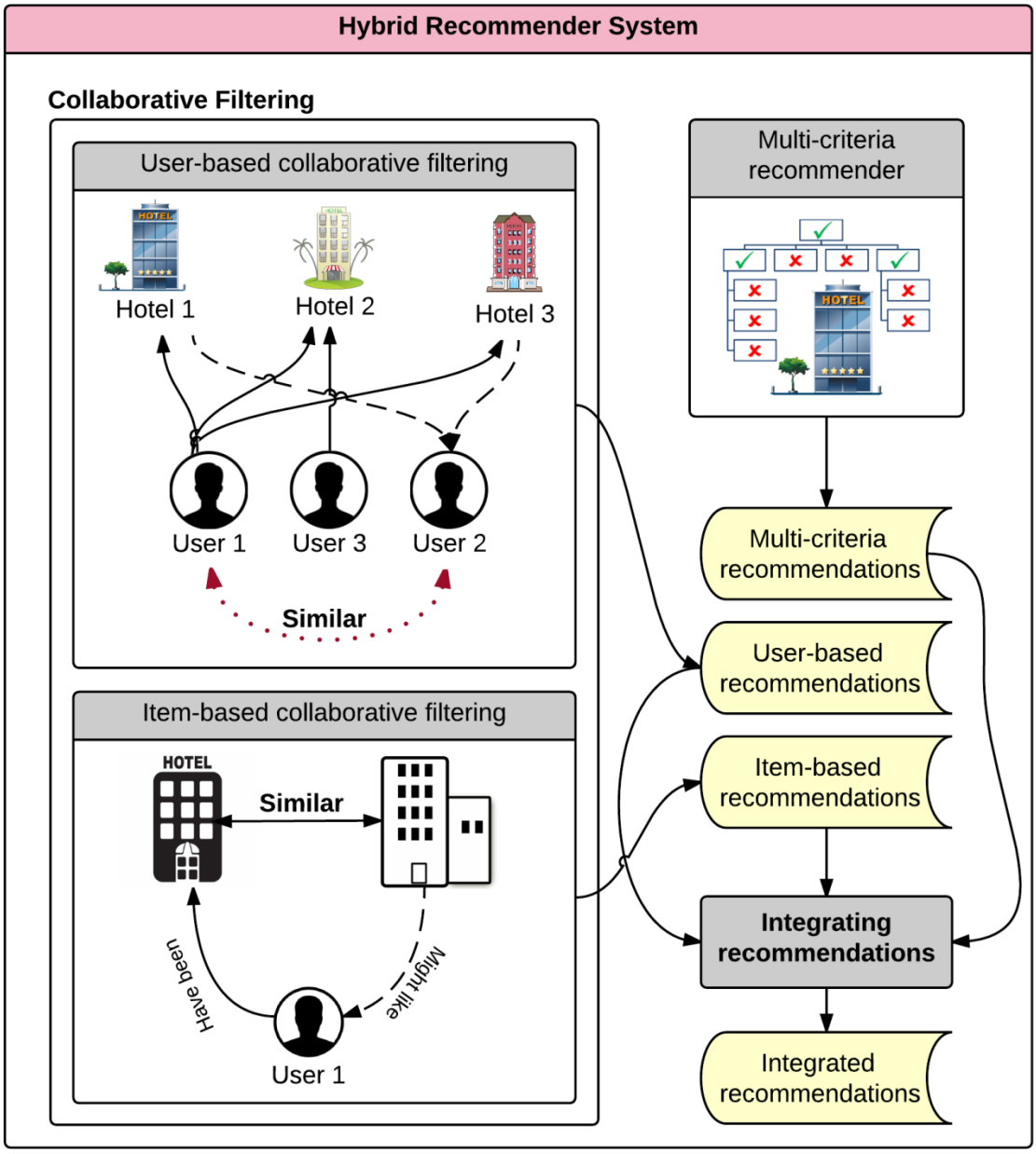

Figure 13. The hybrid recommender system design. The system contains of three main sub-modules: (1) user-based collaborative filtering module, (2) item-based collaborative filtering module, and (3) multi-criteria recommender. The user-based collaborative filtering module recommends items based on users similarities, whereas the item-based module makes recommendations based on the items similarities. The multi-criteria recommender considers various multi-aspect ratings on hotels for making the recommendations. The hybrid recommender system produces four different outputs, i.e. recommendations

made by each of the mentioned sub-modules along with an integrated composite set of recommendations which is made by combining all the three previously stated recommendations.

4.2.3.3.1 User-Based Collaborative Filtering Module

In user-based collaborative filtering approach, the recommendation is made based on the similarities among users. The assumption here is that users with similar features and/or profiles might share similar interests. To measure the similarities among users, various similarity measures were tested[28]. The *cosine similarity* was found to be the best and most robust similarity measure. Thus, the cosine similarity was used in the user-based collaborative filtering module for calculating the similarity among users. The cosine similarity between two vectors (between two users feature vectors in the hotel recommendation case) is calculated based on the cosine of the angle between them. As the cosine similarity metric can be regarded as an orientation proxy rather than the magnitude[29], it is a good match for finding the users similarities. Theoretically, the cosine similarity between vectors U and V is calculated by the Equation (12) below.

$$\cos\theta = \frac{\vec{U}.\vec{V}}{||\vec{U}||\,||\vec{V}||}. \qquad (12)$$

Using the cosine similarity measure, this module makes recommendations in two steps: (1) Identifying users who share the same rating pattern with the given user, and (2) Using the ratings from the similar users who were found in step (1) for predicting the ratings for the given user. The rating is only predicted for the hotels that the given user has not already been there. The hotels (items) are sorted based on their score, and the top hotels in the sorted list are recommended.

4.2.3.3.2 Item-Based Collaborative Filtering Module

The item-based collaborative filtering is a model-based algorithm. In this module, the similarities among various hotels are calculated based on the cosine similarity measure, as was explained in the previous section. Using the calculated similarities among the hotels (items), rating predictions are made for <user, hotel> pairs that are not present in the dataset, *i.e.* this module recommends hotel to a user that have not been already seen by him/her. Similarities

[28] In particular, Euclidean distance and Manhattan distance were also tested where we found the best result for the cosine similarity measure.
[29] It can be seen as a comparison between users in a normalized feature space.

between two given hotels are calculated using all the users who have rated both the hotels. After modelling the data using the cosine similarity measure, the weighted sum approach is taken for predicting the rating for any (unseen) <user, hotel> pair. For this purpose, all the hotels that are similar to the candidate hotel are first selected, forming the set of *similar hotels*. From the similar hotels, the algorithm selects the ones that have been already rated by the given user. The user's ratings for each of these found hotels are weighted, using the similarity between that hotel and the candidate hotel. Finally, the predictions are scaled by the sum of similarities, and the top hotels are recommended. Item-based filtering approach is considerably faster than the user-based collaborative filtering, however, the item similarity table should be maintained and updated over time. The item-based collaborative filtering module not only provides rating predictions on hotels that have not been already seen by a user, it can be also easily set to act as an independent recommender system in high-traffic times of the system.

4.2.3.3.3 Multi-Criteria Recommender Module

To overcome the limitation of the single criterion value, *i.e.* the overall rating, a multi-criteria recommendation engine is implemented in this module. This can help to improve the quality of the final recommendations through representing more complex preferences of each user, as the suitability of the recommended hotel for a given user might depend on more than one rating aspect. The multi-criteria recommender module is provided with multi-criteria rating data, *i.e.* users ratings on multiple and different aspects of the hotels. For example, let us suppose we have a two-criterion hotel recommender system where the data contain users' preferences on two features of a hotel such as location, and quality of the service. A user may like the location, but not satisfied with the quality of the service offered in a hotel, *e.g.* $R(u, h) = (5, 1)$. If we just simply apply the same weights on the two ratings, that is taking an average rating, for making the recommendations, we will obtain 3 out of 5 as user's overall satisfaction in the single-rating setting. However, different situations might lead to the overall satisfaction of 3, such as (5,1), (3,3), or (2,4), *etc*. Thus, although the overall satisfaction in this recommendation setting might be equal, two users might have completely different rating patterns on each criterion of a hotel. This additional information on each user's preferences might help to improve the model accuracy and capability in learning users' preferences. For this purpose, new recommendation techniques,

rather than the traditional methods, should be implemented and employed in order to take advantage of this additional meta-data.

Extending recommendation techniques to be able to handle multi-criteria ratings has been one of the hot topics in recommendation systems community within the past decade. Theoretically, the utility based formulation of multi-criteria hotel recommendation problem can be stated as in Equation (13).

$$R: Users \times Hotels \rightarrow (R_1, R_2, \dots, R_k). \qquad (13)$$

To address the multi-criteria recommendation problem, we considered user-based multi-criteria collaborative filtering approach. The architecture is almost the same as the user-based collaborative filtering as explained earlier in section 4.2.3.3.1. The only difference is in defining and calculating the similarity measure in order to stand for the availability of multi-criteria information. There exist a number of studies that focused on extending the traditional similarity measure calculations to reflect multi-criteria information (Adomavicius & Kwon, 2007; Manouselis & Costopoulou, 2007). One common approach is to aggregate the traditional single-criteria similarities. In this thesis, the cosine similarity calculation was modified to reflect the multi-criteria information, and was used in a user-based collaborative filtering module.

In particular, the similarity between any two given users was calculated based on each individual criterion, let us say *k* criterions, using the single criterion cosine similarity measure, as stated in Equation (12). Then, the final similarity between the two given users was calculated by averaging the calculated *k* similarities (Adomavicius & Kwon, 2007). This aggregated users' similarity measure was then used by a user-based collaborative filtering module to make the recommendations.

4.2.4 Performance Evaluation

Evaluating the performance of a machine learning system is a crucial task for validating the results as well as comparing it with the other similar solutions. There exist so many performance metrics in the literature as they constitute a separate field of research. Since performance measures are often complex and difficult to interpret, selecting the most proper measures for a problem is also a critical issue. In addition, performance measures should be

selected wisely in accordance with the subject problem, *i.e.* not all the performance metrics are useful in all research projects. For example, in a health care setting, if a machine is supposed to identify patients at high risk, the rate of true positives would be of high importance. Moreover, various validation strategies can be employed for validating the results.

In this research, the leave-one-out cross validation (LOOCV) strategy was selected for validating the results. In LOOCV with *n* data points, 1 observation (data point) is considered as the validation set in each run, while the remaining data points form the training set. The procedure is repeated *n* times, taking all data points as the validation set once. LOOCV can be very time consuming for big *n*s, as the procedure requires to learn and to validate *n* times. The LOOCV procedure is depicted in Figure 14.

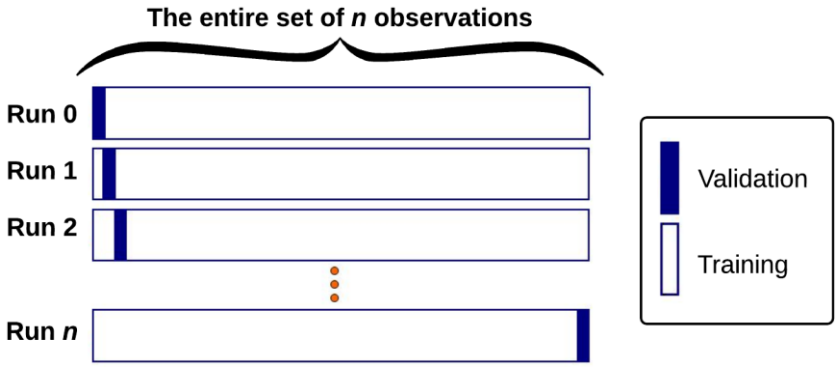

Figure 14. Leave-one-out cross validation design used for testing the performance of the proposed recommender system and validating the results.

Several performance measures, *i.e.* accuracy, specificity, sensitivity, *etc.*, as well as some error measures, are calculated for evaluating the proposed recommendation system. Accuracy evaluates how close the recommendation results are to the users' preferences. It is one of the simplest and most important performance metrics for evaluating the quality of recommendations, and is defined as: $(\mathrm{TP}+\mathrm{TN})/((TP+FN)+(TN+FP))$, where *TP* is number of true positives, *TN* is the number of true negatives, *FN* is the number of false negatives, and *FP* is the number of false positives. Recall, also called true positive rate or *sensitivity*, measures the proportion of correctly identified positives, *i.e.* number of recommendations that were suggested correctly based on the test data. Thus, recall is calculated as: $\mathrm{TP}/(TP+FN)$. In other words, in the case of hotel recommendation, accuracy can be regarded as a ratio which indicates how precise the recommendations are, and recall can be considered as how complete the

recommendations are. *Specificity*, or true negative rate, measures the proportion of correctly identified negatives. In the case of hotel recommendation, specificity measure shows the percentage of hotels that have not been recommended to the users correctly, due to their specific characteristics and preferences. In other words, specificity measures the avoiding of false negatives, whereas sensitivity quantifies the avoiding of false positives. In this sense, a hotel recommender system that is 100% sensitive, *i.e.* all user-hotel preferences are identifies, and 100% specific, *i.e.* no improper hotel is recommended to users, can be regarded as a prefect predictor. However, in real-life problems, any predictor or recommender system may have an error bound in predicting the target variable(s). *Informedness*, is another performance metric that will be calculated to assess the proposed recommender system. Informedness is defined as: (*Specificity* + *Sensitivity* – *1*), and evaluates the probability of an informed decision, in this case recommendation. Cases with informedness metric closer to 1, represent more appropriate use of information, and cases closer to 0 represent chance-level performance (Powers, 2011).

Apart from the decision-based metrics that will be calculated, and were explained earlier, a number of prediction-based measures are also calculated. Mean absolute error (MAE), as stated in Equation (14), is a prediction-based metric that will be calculated. MAE is a metric that measures how close the predictions, *i.e.* predicted ratings in our case, are to the actual outcomes, *i.e.* the exact ratings.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|r_i' - r_i|. \qquad (14)$$

In Equation (14), n is the number of observations, r_i' is the i^{th} predicted rating, and r_i is the actual i^{th} rating. In other words, MAE is the average of the absolute error values. The next measure, is mean squared error (MSE), as defined in Equation (15). MSE is another measure for assessing the quality of the predictor, and calculates the average of squared error deviations.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(r_i' - r_i)^2. \qquad (15)$$

And finally, root-mean-square error (RMSE) is calculated which is an alternative measure for assessing the differences between the actual values and the ones predicted by an

estimator. RMSE is obtained by taking the square root of MSE, thus, RMSE can be regarded as the square root of the sum of variance and squared bias, also known as the standard deviation. Both MAE and RMSE measure how close the estimates are to the actual values.

5 RESULTS

Recommendation systems are now being widely used in industry, and have been an established field of research for more than 10 years, especially within the interactive media research. They serve as the basis of many successful commercial platforms, providing users with personalized and intelligent recommendations, aiming to boost the profit margins of companies. In the previous section, an intelligent hybrid recommender framework was proposed for the hotel recommendation problem. The proposed system, intelligently leverages form various sources and types of data to tailor the recommendations for the users. In this section, the performance evaluation results of the proposed framework, including the evaluation of all the sub-recommenders as well as the composite set of recommendations, are presented and discussed in details.

5.1 User-Based Collaborative Filtering

As explained in section 4.2.3.3.1, the user-based collaborative filtering module incorporates various features of different types, including content features, collaborative filtering features, and extracted features, to recommend the most proper hotels to any given user. The input to this sub-system is the selected cluster for the given user, as describes earlier in section 4.2.3.1. Thus, the module is provided with the selected data cluster which contains records that are most similar to the given user, and applies user-based collaborative filtering technique to recommend hotels.

To test the accuracy and performance of the user-based collaborative filtering module, we used leave-one-out cross validation technique (LOOCV, as describes in section 4.2.3.3.3). Furtermore, cosine similarity was used for measuring the distance between different users. In LOOCV, a model is fit on all the data points except one, and the left out data point is used to calculate the model prediction error. This procedure is repeated for all the data points in the dataset, taking each one once as the left out data point. Therefore, the model was fitted *n* separate times, where *n* is the number of data points (in each cluster).

Figure 15 depicts the prediction-based metric for the user-based collaborative filtering module. As explained earlier, all the reported statistics, *i.e.* MAE, MSE, and RMSE, compare the actual ratings with their estimates, but in a slightly different manner. As seen in Figure 15, the

user-bases CF is able to predict the ratings for user-hotel pairs with relatively small error. According to MAE, the average magnitude of the errors in the prediction set, regardless of their direction, is relatively small, with the value of 0.65. This can be also regarded as a sign of high accuracy of the user-based CF sub-system that will be discussed in more details later in this section. In other words, the average of the absolute values of differences between the predicted ratings and the corresponding observations over the entire LOOCV validation procedure is slightly higher than half a scale, indicating the high performance of the sub-system. One should note that the error rate is promising considering the large data size that was used. MSE and RMSE are also positive numbers, ranging from 0 to ∞, where lower values indicate higher accuracy. RMSE is a quadratic error metrics which gives higher weights to larger errors, as the errors are squared before they are averaged. Thus, RMSE can be of special interest in this research, as large rating prediction errors are not desirable since they might lead to wrong recommendations. According to the results, RMSE of the user-based CF module is also relatively small, with the value of 1.16. It should be noted that RMSE is always greater than or equal to MAE. Here, the difference between RMSE and MAE is approximately equal to 0.5, *i.e.* $1.16 - 0.65 = 0.51$, which indicates that the variance in the individual errors is also relatively small. MSE is the second moment of the error, and it takes both the variance of the estimator and its bias into the account. If MSE equals to zero, then the model is 100% accurate. For this specific problem, it would be suggested to use MSE measure to compare the accuracy of different sub-systems. However, from Figure 15, it can be seen that the mean-square error of the user-based CF module is relatively small, as well.

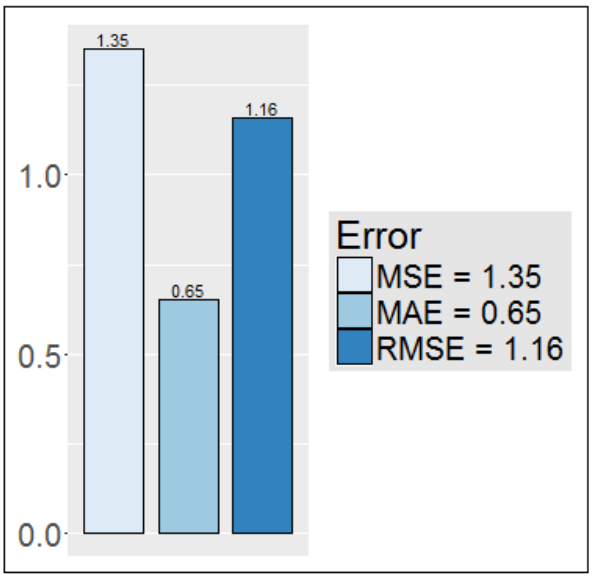

Figure 15. Prediction-based metrics for the user-based collaborative filtering module. According to the measures, the user-based CF module is able to predict the ratings for user-hotel pairs with high accuracy. The small difference between RMSE and MAE measures also indicates that the variance in the individual errors is relatively small.

We further investigated the performance of the user-based collaborative filtering module by calculating a number of decision-based metrics, *i.e.* accuracy, sensitivity, specificity, and informedness (Figure 16). As seen, the module is more than 83% accurate in predicting the right hotels to be recommended to the users. This confirms that the module is able to make robust, accurate, and acceptable predictions through learning the users' preferences accurately. However, analyzing the other measures is necessary to guarantee that the model is well fitted and good enough for the subject problem. Sensitivity is number of true positives divided by true positives and false negatives. It indicates the number of positive predictions divided by the number of all possible positives in the data. According to the results, the user-based module is highly sensitive, *i.e.* 94.1%, indicating that the model is highly complete, capturing almost all the positives in the data. Thus, the model is able to effectively learn users' preferences.

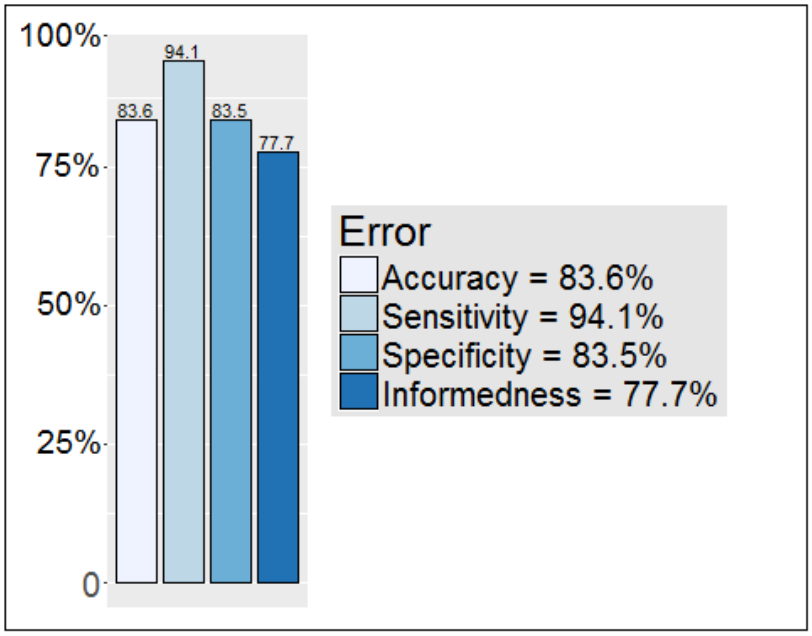

Figure 16. Decision-based performance metrics, calculated for the user-based collaborative filtering module. The module is highly accurate (83.6%) in predicting the user-hotel preferences and recommending correct hotels to the users. According to the specificity and sensitivity measures, the model is effectively learning user preferences, capturing both true positives and true negatives. The informedness measure also shows that the system is making informed decisions in recommending hotels to the users.

In addition, the specificity of the model is also considerably high (83.5%), as seen in Figure 16. Specificity can be regarded as the effectiveness of the system in identifying true negatives. From the calculated specificity and sensitivity of the model, it is clear that the model is able to learn the preferences, correctly identifying both desirable and undesirable items. This is of high importance in travel recommendation systems, as unintelligent recommendations might cause users to even leave the system. Moreover, from the informedness measure (77.7%), it can be said that the module is appropriately using the information hidden in the data to make informed decisions in recommending hotels to similar users in the system. This reconfirms that the recommendations, and as a result the other performance measures, are not obtained randomly and by chance.

5.2 Item-Based Collaborative Filtering

This module accounts for the question of "which hotels are similar to each other?". Item-based collaborative filtering helps the proposed hotel recommendation systems, at least, in two ways: 1) By recommending items (hotels) to users that have not already been rated by them,

thus, solving the problem of new items to users, and 2) By improving the overall speed of the system and the possibility of acting as a fast independent recommender module, if necessary. In very large scale datasets in real-life situations, calculating user similarities might harm the response rate and the speed of the system. In such cases, the implemented item-based collaborative filtering can give better results through calculating the items similarities in advance, so that a user can receive recommendation faster. That is, although in item-based collaborative filtering it is required to examine all the data, the comparisons among items (hotels) will not change as frequent as the ones between users. Thus, it is not needed to calculate each hotel's most similar hotels continuously, and the calculation load can be forwarded to low-traffic times.

As mentioned in section 4.2.3.3.2, the item-based collaborative filtering module was designed such that it recommends unseen hotels that are similar to the ones that have been already rated by a user. Thus, this module was designed such that it acts as a complementary component to the other modules in the recommender engine, recommending unseen hotels. Therefore, it is not possible to calculate the accuracy of this sub-module at the current setting, as the data are not available. However, in operation, several strategies can be taken by the operating website in order to investigate if the users have welcomed the new hotel suggestions. This can be done, for example, by incorporating web cookies in order to capture user navigation traces and behaviors, or to obtain users' feedbacks on new recommended items.

5.3 Matrix Factorization

As described earlier (refer to Section 4.2.3.2), the matrix factorization module approximates the missing rating values. The module was designed as part of the hybrid recommender system to account for the new to the system hotels, as they cannot be perfectly identified and recommended by collaborative filtering based approaches. Since the user-hotel rating is a non-negative matrix, the non-negative matrix factorization (NMF) approach was implemented. And, since the user-hotel rating matrix was very sparse, the nonnegative double singular value decomposition (NNDSVD) approach was used for initialization. NNDSVD which is a method for enhancing the initialization stage of the NMF approach is proven to be very effective for rapid reduction of NMF algorithm estimation error (Boutsidis & Gallopoulos, 2008).

One should note that the performance of the module and the precision of the predicted user-hotel ratings are highly dependent on the number of the decomposed matrices, and as a result their dimension. That is, small vectors might not possess enough explanatory power to distinguish between various items or users, while large vectors might cause over-fitting problem. Although in NMF approach the given matrix, here the user-hotel rating matrix, is usually factorized into two matrices, we further checked for the best number of components for the subject problem. That is, to check the performance of the matrix factorization module, a set of prediction-based metrics, similar to the ones used in Section 5.1, was evaluated versus different number of components. The experiments helped to improve the accuracy of the system through analyzing various reduced user-hotel vectors to set the best number of components for the given problem. As can be seen in Figure 17, although the curve exhibits slight fluctuations, as expected, the root mean square error of the NMF module is minimized when the number of components is set to 2.

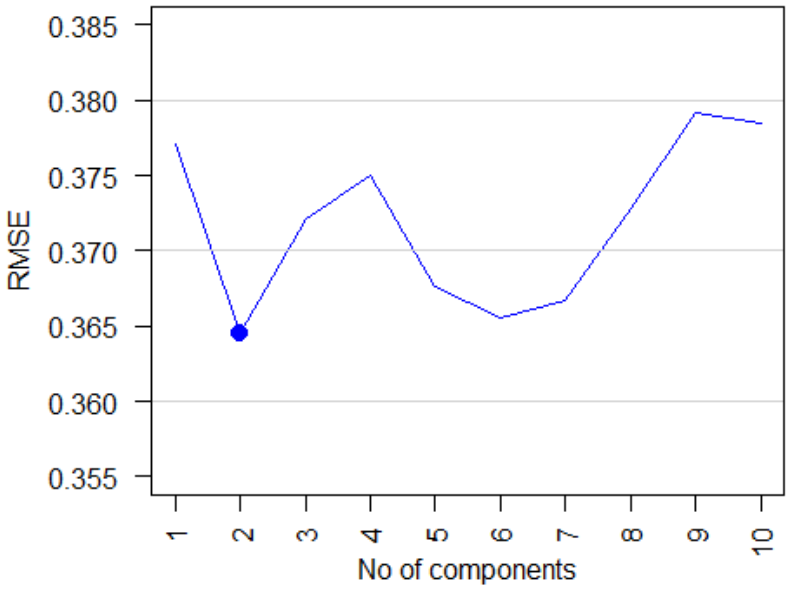

Figure 17. NMF module root mean square error (RMSE) versus number of components. As highlighted by a blue dot on the figure, RMSE is minimized when the user-hotel rating matrix is factorized into two matrices.

Having set the number of components, the number of iterations was also recorded to find the best parameter such that the error is minimized. As seen in Figure 18, RMSE of NMF module remains constant after 24 iterations. Thus, the maximum number of iterations was set to 50 to guarantee that the model iterates enough in a way that the error is minimized.

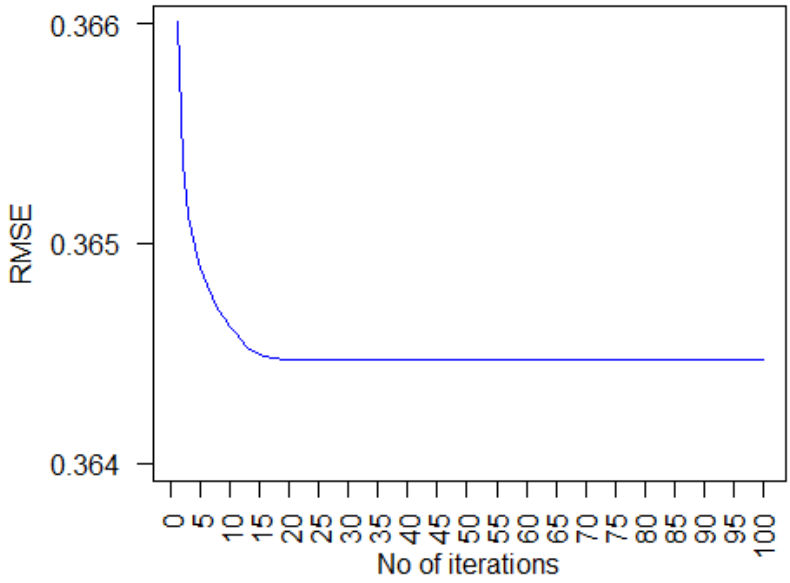

Figure 18. NMF module root mean square error versus number of iterations. As seen, RMSE remains constant after ~24 iterations.

Having found the best performing parameters, a set of other prediction-based performance metrics were calculated to better evaluate the NMF module (similar to the approach taken in the previous section). As seen in Figure 19, RMSE of the NMF module is considerably small, with the value of 0.364, and is larger than MAE and MSE as expected, according to the definition of the mentioned measures. Based on the observed measures, *i.e.* MAE, MSE, and RMSE, it can be said that the NMF module is highly accurate in decomposing the user-hotel rating matrix and predicting the ratings for any user-hotel pair. Thus, NMF module can be employed with high accuracy to recommend (unseen to the user or new to the system) hotels to the users. In the next section, the performance of the multi-criteria recommender module is examined.

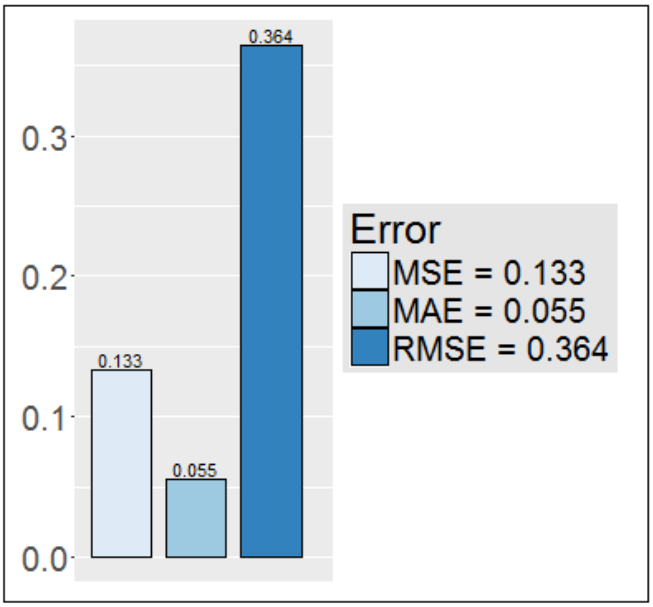

Figure 19. Prediction-based metrics for NMF module. According to the measures, the NMF module is able to generate the original matrix and predict the ratings for (unseen) user-hotel pairs with very high accuracy.

5.4 Multi-Criteria Recommender

Multi-criteria recommender systems have been proven to have the potential to facilitate the recommendations in several application domains. In this thesis, a multi-criteria recommender module was designed and implemented which is able to leverage from explicit and implicit extracted features in order to enhance the recommendations quality. This section is dedicated to the performance analysis of the designed module, as explained earlier in section 4.2.3.3.3. Here, the user-hotel rating is not a matrix anymore, as it takes the form of a tensor since multi-criteria rating system is used. For the purpose of evaluation, two separate scenarios are investigated and discussed here: **Scenario-1**) The rating vector for each user-hotel pair contains ratings on three separate aspects, *i.e.* the worthiness of the hotel against the paid amount, the quality of the service, and the implicit score that was obtained as the polarity of the users' reviews on hotels[30], and **Scenario-2)** The rating vector contains ratings on five different aspects namely worthiness of the hotel against the paid amount, quality of the hotel rooms, location of the hotel, cleanness of the hotel, and quality of the service. In the second scenario the polarity rating is excluded.

[30] To see the details of the model, please refer to section 4.2.1.

5.4.1 Scenario-1, Implicit and Explicit Ratings

In this scenario, performance of the MCR module is checked for the case that implicit and explicit feedbacks are used to assess different aspects of the hotels and form the rating vectors for user-hotel pairs. For this purpose, the leave-one-out cross validation technique (LOOCV, as explained in section 4.2.3.3.3) was employed to test the accuracy and performance of the system. The cosine similarity was used for measuring the distance between users with regard to each aspect, and the similarity scores were aggregated to create a single similarity measure[31]. Since LOOCV was used, the model was fitted and tested *n* times, where *n* is the number of data points, and the model prediction error and performance metrics were calculated and averaged over the *n* iterations.

The results of the prediction-based metrics are shown in Figure 20. As seen, MAE metric that is a measure for the average magnitude of the errors in the predictions set without considering their directions, is considerably low. Since the MSE and RMSE scores are also significantly low, it can be said that the module is highly accurate in predicting the ratings for user-hotel pairs. Comparing the results with Figure 15, it is observed that the multi-criteria recommender system is performing better than the user-based collaborative filtering module. Although MAE of the multi-criteria recommender is slightly higher, i.e. 0.76 vs. 0.65, since the other metrics are much better for the multi-criteria system, it can be said that the multi-criteria recommender is able to benefit from the multi-aspect evaluations to enhance the quality of the recommendations. Moreover, the very small difference between MAE and RMSE in the multi-criteria recommender indicates that the variance in the individual errors is considerably small, even better than the user-based collaborative filtering system. That is, the model is making relatively small errors in predictions.

[31] For further explanation, please refer to section 4.2.3.3.3.

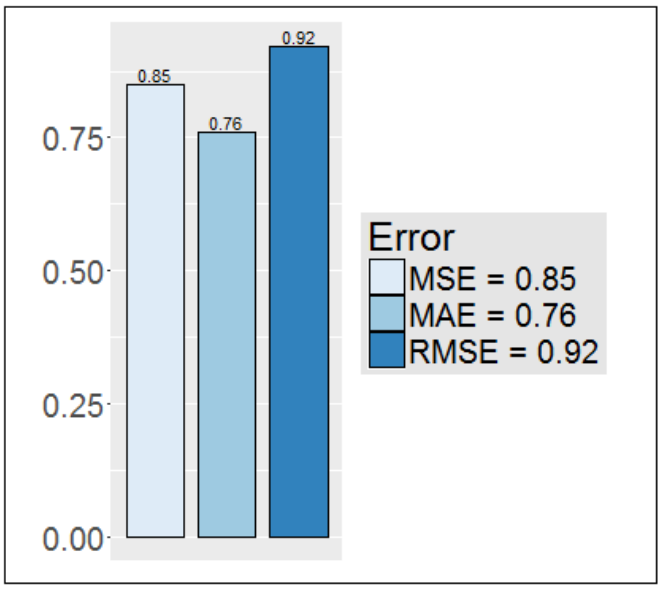

Figure 20. Prediction-based metrics for the MCR module, using implicit and explicit ratings. According to the measures, the MCR module is able to incorporate all the given aspects of ratings and to predict the ratings for user-hotel pairs with high accuracy. The small difference between RMSE and MAE measures also indicates that the variance in the individual errors is considerably small.

To further investigate the performance of the multi-criteria recommender module, a set of decision-based metrics were calculated. As seen in Figure 21, the module is more than 90% accurate in predicting the right hotels to recommend to the users. This confirms that the module is able to make robust, accurate, and acceptable predictions. As expected, this is higher than the accuracy of the user-based collaborative filtering module (Figure 16). Analyzing the other measures also confirms that the multi-criteria recommender model is well fitted for the subject problem. That means, the multi-criteria recommender module is very sensitive, *i.e.* 91%, indicating that the model is highly complete, capturing almost all the positives in the data. Thus, the model is able to effectively and (almost) completely learn users' preferences. Moreover, the specificity of the model is also significantly high, exceeding 90%. Therefore, the module can effectively identify true negatives. According to the specificity and sensitivity metrics, the model is highly capable of learning users' preferences such that desirable and undesirable items are correctly identified. Finally, the informedness measure (81.1%) shows that the module is appropriately detecting the hidden information in the data and the patterns of preferences, and employs them to make informed decisions in recommending hotels to similar users in the system.

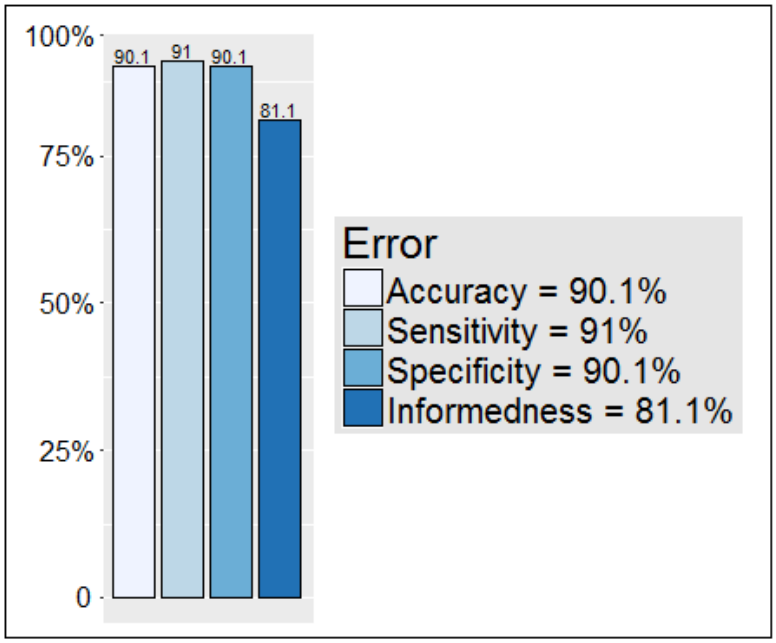

Figure 21. Decision-based performance metrics, calculated for the multi-criteria recommender module. The accuracy of the module in predicting the user-hotel preferences and recommending correct hotels to the users exceeds 90%. According to the specificity and sensitivity measures, the model is highly potential in learning user preferences, capturing both true positives and true negatives. The informedness measure also indicates the high ability of the system in making informed decisions.

5.4.2 Scenario-2, Multi-Aspect Explicit Ratings

In the second scenario, the same performance evaluation approach as the first scenario is performed. The only difference is in the rating vectors where here the implicit rating, *i.e.* users' reviews polarity, is excluded. In addition, the rating vector contains five different aspects in comparison with three in the first scenario. To check the performance, prediction-based and decision-based metrics are used within a LOOCV module. And, cosine similarity is employed for calculating the distance between users.

Figure 22 depicts the calculated prediction-based metrics. As observed, all the measures are larger than the ones for the first scenario (Figure 20). That means the MCR module performs better when both implicit and explicit ratings are used. MAE, which is expectedly the smallest error metric, is almost equal to 1, meaning that MCR module is able to predict the ratings with ~1 unit error in rating. RMSE and MSE are equal to 1.39 and 1.92 respectively. The considerable difference between MAE and RMSE in the second scenario indicates that the variance in the individual errors is not very small.

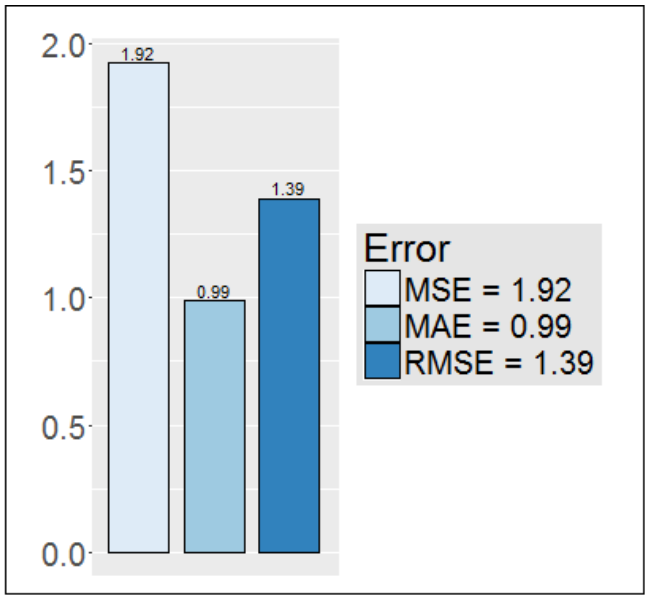

Figure 22. Prediction-based metrics for the MCR module, using only explicit multi-aspect ratings. According to the measures, the second scenario shows lower performance in comparison with the first one where the implicit rating was also included. The difference between RMSE and MAE measures indicates that the variance in the individual errors is not very small.

Analysis of the decision-based performance metrics (Figure 23) reveals that the module is slightly less than 90% accurate in predicting the right hotels to recommend to the users. As expected, this is lower than the accuracy of MCR module in the first scenario (Figure 21). The same is valid for the other error metrics such that it is confirmed that the implicit ratings contribute to the improvement of the recommender system. Based on the observed measures, although the model is less accurate than the first scenario, the specificity and sensitivity measures are still considerably high, thus, indicating that the model is highly potential in learning user preferences and capturing both true positives and true negatives. The informedness measure also shows that the model is not predicting ratings randomly.

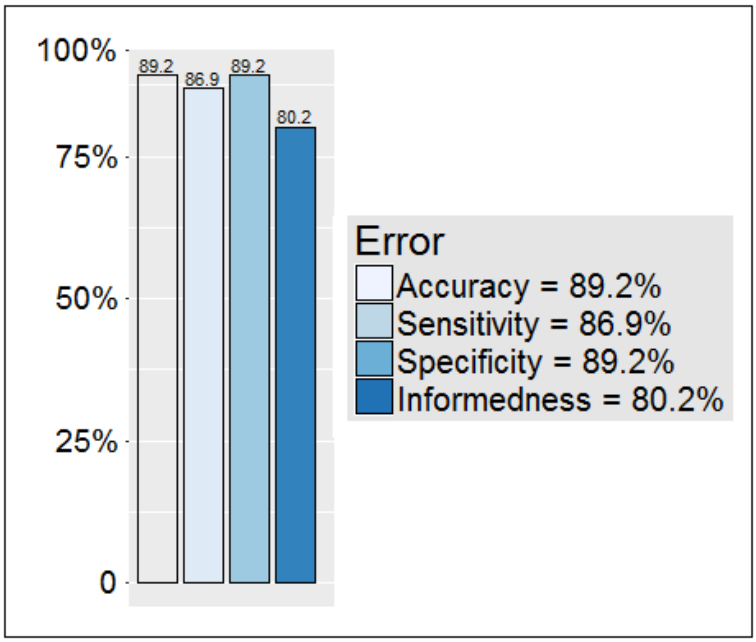

Figure 23. Decision-based performance metrics, calculated for the MCR module, using only explicit multi-aspect ratings. The accuracy of the module in predicting the user-hotel preferences and recommending correct hotels to the users is ~ 89%. According to the specificity and sensitivity measures, the model is very potential in learning user preferences, capturing both true positives and true negatives. But, it performs slightly worse than the first scenario where implicit and explicit ratings were used. The informedness measure also indicates the high ability of the system in making informed decisions.

5.5 The Composite Set of Recommendations

Having evaluated the performance of the individual components of the hybrid recommender systems, in this section, accuracy of the composite set of recommendations is evaluated. In creating the composite set of recommendations, without losing the generality, higher weight is given to the modules with higher performance. This reflects the common human behavior in giving more weights to more important (more reliable) persons. The maximum size of the composite set of recommendation is considered to be 10, containing (maximum) 10 different hotels to be recommended to each user[32]. However, the system can be easily adjusted to provide the user with a larger/smaller set of recommendations. To check the performance, the item-based and NMF recommendations are excluded as these modules are used for recommending new to the system or unseen hotels to the users, as explained in sections 4.2.3.2 and 4.2.3.3.2.

[32] Out of the 10 recommendations, 5 is from the MCR module with implicit and explicit ratings, 3 from the user-based collaborative filtering module, and 2 from the MCR with multi-aspect explicit ratings.

In this experiment, the predicted ratings are compared with the actual *total* ratings. One should note that evaluating recommender systems is extremely difficult. Here, the total rating, which is the single rating that a user gives to a hotel, is considered as the ground truth. One other way is to compare the predicted rating for each sub-module with the actual ratings in that specific sub-module. In this scenario, the accuracy measures would be the ones for each sub-module, averaged over the sub-modules. Same as the approach in the previous sections, LOOCV is used for calculating prediction-based and decision-based performance metrics.

The results of the prediction-based metrics calculations for the composite set of recommendations are depicted in Figure 24. As seen, the system performs reasonably well in predicting the ratings for user-hotel pairs. The relatively small difference between RMSE and MAE confirms the existence of low variance in the individual errors. While indicating there is some variation in the magnitude of errors, it also confirms that large errors are very unlikely. According to the results, the average difference between the predicted rating based on the composite set of recommendations and the observed total rating is 0.78. This highlights the power of the proposed system in learning the users' preferences. Interestingly, the accuracy of the composite set of recommendations exceeds 98%.

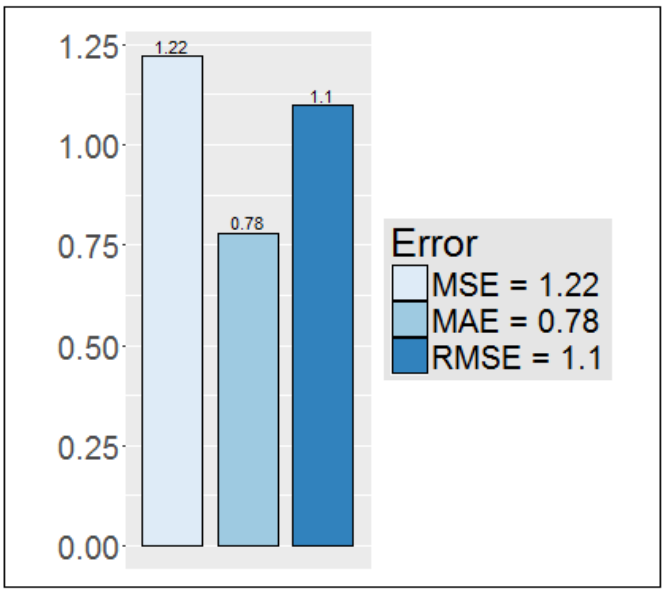

Figure 24. Prediction-based metrics calculated for the composite set of recommendations. The results again confirm the high performance of the proposed system. The relatively small difference between RMSE and MAE measures indicates that the variance in the individual errors is small. However, the difference itself declares that there is some variation in the magnitude of the errors, although large errors are very unlikely to have occurred.

6 CONCLUSIONS

In this thesis, a novel hybrid solution was proposed for predicting ratings for user-hotel pairs and making the recommendation. The proposed approach combined collaborative filtering with matrix factorization and clustering techniques to improve the performance. Moreover, users' text reviews were converted to polarity scores, reflecting implicit feedbacks, and were integrated into the feature space. In addition, topic modelling techniques were applied to generate implicit features from users' reviews, reflecting unique points of interests for each user in the system. The diversity of the features types, including both implicit and explicit feedbacks, as well as the integrity of the techniques, helped the system to reach outstanding accuracy and performance. Although there are hybrid recommendation designs in the literature, to the best of our knowledge, this is the first one in hotel recommendation domain that applies a triangulation technique and incorporates comprehensive sentiment analysis module and keyword extraction techniques to obtain content information and use them along with a diverse set of other features to solve the problem. The main advantages of the proposed design are: 1) A hybrid design which is well suited to the subject problem and can be operated easily, 2) Highly accurate predictions, 3) Use of implicit and explicit feedbacks and the novelty in employing sentiment analysis and keyword extraction techniques for extracting new features, 4) The system's ability in recommending "new to the user" items as well as "unseen" ones, and 5) Benefitting from a multi-criteria rating system that helped the recommender engine to better learn users' preferences.

The entire system was implemented in Python programming language. The required data were collected from multiple sources and then integrated into a single MySQL database. In addition, users' complementary information was also retrieved from the internet and integrated into the database, after matching the entities. LOOCV approach was used for testing the performance of the proposed system, as well as all the sub-systems. That is, various experiments and evaluation metrics were used to demonstrate the high performance of the hybrid recommendation system.

To speed up the system as well as to improve its accuracy and scalability, clustering techniques were applied on users vectors, grouping them into various clusters based on their characteristics. This also played an important role in improving the recommender system

performance in comparison with the basic collaborative filtering algorithms. For this purpose, the system employed various types of content features such as user's age, and location. Moreover, matrix decomposition techniques were employed to solve the cold start problem, making the system capable of drawing inferences for the new to the system items about which it has not yet enough information. The multi-criteria recommender module also empowered the system with multi-aspect ratings that enabled it to provide more accurate recommendations. In addition, this thesis presents an innovative technique for extracting implicit features and converting them to an implicit rating score. The use of implicit features here was found to be crucial, as it was observed that incorporating them augments the system performance through providing it with deeper understanding of user preferences and characteristics.

Comparing the results with the literature, it was observed that the sentiment analysis module shows promising performance in predicting the sentiment of a given sentence/text, with 85% accuracy. This is higher than several similar studies such as Pak and Paroubek (2010) and Rosenthal *et al.* (2015)[33]. The proposed recommender framework also performs more effectively than the approaches in the literature (Table 3). For example, Nilashi *et al.* (2015) proposed a multi-criteria recommender system for tourism domain for which they obtained MAE of 0.86, and compared their results with several other algorithms, using TripAdvisor data. According to their findings, mean absolute error for the standard collaborating filtering method (Adomavicius & Kwon, 2007) equals 1.37. MAE was found to be 1.28 for Total-Reg algorithm (Adomavicius & Kwon, 2007), and, 0.89 for ANFIS and HOSVD algorithm (Nilashi *et al.*, 2014). As seen, the recommender engine that was proposed in this thesis outperforms the similar available systems.

Table 3. Comparing the performance of the proposed hotel recommender system with similar existing systems in the literature

Recommender	Reference	Dataset	MAE
Standard collaborative filtering	Adomavicius & Kwon (2007)	TripAdvisor	1.37
Total-Reg	Adomavicius & Kwon (2007)	TripAdvisor	1.28
ANFIS and HOSVD	Nilashi *et al.* (2014)	TripAdvisor	0.89
Clustering and PCA-ANFIS	Nilashi *et al.* (2015)	TripAdvisor	0.86
Our proposed recommender	Ebadi & Krzyzak (2016a)	TripAdvisor	**0.78**

[33] In this survey, performance results of 11 different systems in predicting phrase-level binary polarity of tweets were listed, where all the systems have accuracy lower than 85%.

In general, it was observed that the proposed solution can fit well with the specified problem, and is well-tailored for the target business, covering all the aspects that might be important in a real-life business case. It is also flexible enough to take the speed-accuracy trade-off into the account through giving higher weights to different sub-systems based on the available conditions in the company and the market situation. As discussed earlier in section 5, the system is highly customizable and can be easily adjusted for different scenarios or even different businesses, with some minor changes. However, apart from the system architecture, the error metrics are also required to be selected wisely, in accordance with the business nature and problem objective(s).

7 FUTURE WORK AND LIMITATIONS

Recommender systems are now being used widely in various types of applications and domains. There are a number of traditional ways to measure the effectiveness of recommenders' architecture and performance. However, a precise evaluation of a recommender system is more accessible in an actual situation. Such situation should be properly evaluated providing a clear picture of the domain properties and characteristics, business goals and objectives, and behavior of the algorithm. The proposed system will be soon operated in a start-up company in Montreal. Several complementary measures, such as time that a user spent on a web page, number of hits for recommendations, click tracking, like or dislike for a recommendation, *etc.*, are considered to be incorporated. These measures will definitely provide more flexibility in assessing the performance of the system and its suitability.

Another direction for the future research might be using more data. Although large scale data were used in this thesis for training the recommender modules, more data (from other sources) can be also employed to check the performance of the system. In addition, more features on users and/or hotels can definitely help the system, at least in better clustering of the user which might ultimately lead to higher performance. Moreover, as discussed earlier, the user-hotel-rating matrix is extremely sparse, thus, testing the proposed system on less sparse data can be also suggested. Although multi-aspect rating was also used in this system, more rating data and/or more rating dimension can be also helpful in determining the behavior of the hybrid recommender engine more accurately.

Although a set of content features was used in this thesis for training the recommender systems, there might exists many other (content) features on users or hotels that are worth to examine. One example is using zip codes rather than region/city to group users. This might provide the system with more concrete information that can be used to provide better targeted recommendations. In addition, more data features provide more flexibility which might lead to defining meaningful combinations of various features such as <hotel keyword-user age> and incorporating them into the system, that surely contains valuable information. This, for example, can unveil what particular characteristics in hotels attract different age groups. The combination of the proposed hybrid design with an additional knowledge database will surely result in even better predictions.

REFERENCES

Adomavicius, G., & Kwon, Y. (2007). New recommendation techniques for multicriteria rating systems. *Intelligent Systems, IEEE, 22*(3), 48-55.

Adomavicius, G., & Tuzhilin, A. (2005). 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, 17*(6), 734-749.

Adomavicius, G., & Zhang, J. (2012). Stability of recommendation algorithms. *ACM Transactions on Information Systems (TOIS), 30*(4), 23.

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proc. 20th Int. Conf. very Large Data Bases, VLDB, , 1215* 487-499.

Amatriain, X., & Basilico, J. (2012). Netflix recommendations: Beyond the 5 stars (part 1). *Netflix Tech Blog, 6*.

Angel, A., Chaudhuri, S., Das, G., & Koudas, N. (2009). Ranking objects based on relationships and fixed associations. *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology,* 910-921.

Balabanović, M., & Shoham, Y. (1997). Fab: Content-based, collaborative recommendation. *Communications of the ACM, 40*(3), 66-72.

Bao, Y., Fang, H., & Zhang, J. (2014). Topicmf: Simultaneously exploiting ratings and reviews for recommendation. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence,* 2-8.

Basu, C., Hirsh, H., & Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. *Aaai/iaai,* 714-720.

Bell, R. M., & Koren, Y. (2007). Lessons from the Netflix prize challenge. *ACM SIGKDD Explorations Newsletter, 9*(2), 75-79.

Bennett, J., & Lanning, S. (2007). The netflix prize. *Proceedings of KDD Cup and Workshop* 35.

Bilge, A., & Polat, H. (2013). A scalable privacy-preserving recommendation scheme via bisecting k-means clustering. *Information Processing & Management, 49*(4), 912-927.

Bilgic, M., Mooney, R., & Rich, E. (2004). Explanation for recommender systems: Satisfaction vs. promotion. *Computer Sciences Austin, University of Texas. Undergraduate Honors, 27*

Bishop, C. M. (2006). *Pattern recognition and machine learning* springer.

Blanco-Fernández, Y., Pazos-Arias, J. J., Gil-Solla, A., Ramos-Cabrer, M., López-Nores, M., García-Duque, J., . . . Bermejo-Muñoz, J. (2008). A flexible semantic inference methodology to reason about user preferences in knowledge-based recommender systems. *Knowledge-Based Systems, 21*(4), 305-320.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *The Journal of Machine Learning Research, 3*, 993-1022.

Bollen, D., Knijnenburg, B. P., Willemsen, M. C., & Graus, M. (2010). Understanding choice overload in recommender systems. *Proceedings of the Fourth ACM Conference on Recommender Systems,* 63-70.

Boutsidis, C., & Gallopoulos, E. (2008). SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition, 41*(4), 1350-1362.

Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence,* 43-52.

Brusilovsky, P., Kobsa, A., & Nejdl, W. (2007). *The adaptive web: Methods and strategies of web personalization* Springer Science & Business Media.

Brusilovsky, P., Kobsa, A., & Nejdl, W. (2007). *The Adaptive Web: Methods and Strategies of Web Personalization* Springer Science & Business Media.

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

Burke, R. (2007). Hybrid web recommender systems. *The adaptive web* (pp. 377-408) Springer.

Candillier, L., Meyer, F., & Boullé, M. (2007). Comparing state-of-the-art collaborative filtering systems. *Machine learning and data mining in pattern recognition* (pp. 548-562) Springer.

Canny, J. (2002). Collaborative filtering with privacy. *Proceedings of 2002 IEEE Symposium on Security and Privacy,* 45-57.

Carbonell, J., & Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* 335-336.

Chen, J., Chao, K., & Shah, N. (2013). Hybrid recommendation system for tourism. *2013 IEEE 10th International Conference on E-Business Engineering (ICEBE),* 156-161.

Chen, T., Zhang, W., Lu, Q., Chen, K., Zheng, Z., & Yu, Y. (2012). SVDFeature: A toolkit for feature-based collaborative filtering. *The Journal of Machine Learning Research, 13*(1), 3619-3622.

Clarke, C. L., Kolla, M., Cormack, G. V., Vechtomova, O., Ashkan, A., Büttcher, S., & MacKinnon, I. (2008). Novelty and diversity in information retrieval evaluation. *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* 659-666.

Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., & Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. *Proceedings of ACM SIGIR Workshop on Recommender Systems, 60*.

Cosley, D., Lam, S. K., Albert, I., Konstan, J. A., & Riedl, J. (2003). Is seeing believing? How recommender system interfaces affect users' opinions. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 585-592.

Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., Livingston, B. (2010). The YouTube video recommendation system. *Proceedings of the Fourth ACM Conference on Recommender Systems,* 293-296.

de Spindler, A., Norrie, M. C., Grossniklaus, M., & Signer, B. (2006). Spatio-temporal proximity as a basis for collaborative filtering in mobile environments.

Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., & Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science (JASIS), 41*(6), 391-407.

Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS), 22*(1), 143-177.

Ebadi, A. & Krzyzak, A. (2016a). A Hybrid Multi-Criteria Hotel Recommender System using Explicit and Implicit Feedbacks. *Proceedings of 18th International Conference on Applied Science in Information Systems and Technology (ICASIST 2016), Amsterdam, Aug. 4-5, 2016.*

Ebadi, A. & Krzyzak, A. (2016b). An Intelligent Multilayer Hotel Recommender Systems. Submitted to the *INFORMS Annual Meeting*. INFORMS.

Erosheva, E. A. (2002). Grade of Membership and Latent Structure Models with Application to Disability Survey Data*. Doctoral dissertation,* Office of Population Research, Princeton University.

Fei-Fei, L., & Perona, P. (2005). A Bayesian hierarchical model for learning natural scene categories. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, 2*524-531.

Felfernig, A., & Burke, R. (2008). Constraint-based recommender systems: Technologies and research issues. *Proceedings of the 10th International Conference on Electronic Commerce,* p. 3.

Frias-Martinez, E., Magoulas, G., Chen, S., & Macredie, R. (2006). Automated user modeling for personalized digital libraries. *International Journal of Information Management, 26*(3), 234-248.

Frias-Martinez, E., Chen, S. Y., & Liu, X. (2009). Evaluation of a personalized digital library based on cognitive styles: Adaptivity vs. adaptability. *International Journal of Information Management, 29*(1), 48-56.

Friedman, N., Geiger, D., & Goldszmidt, M. (1997). Bayesian network classifiers. *Machine Learning, 29*(2-3), 131-163.

Funk, S. (2006). *Netflix Update: Try this at Home,* Blog post: http://sifter.org/~simon/journal/20061211.html.

Gavalas, D., & Kenteris, M. (2011). A web-based pervasive recommendation system for mobile tourist guides. *Personal and Ubiquitous Computing, 15*(7), 759-770.

Ge, Y., Liu, Q., Xiong, H., Tuzhilin, A., & Chen, J. (2011). Cost-aware travel tour recommendation. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 983-991.

Gemulla, R., Nijkamp, E., Haas, P. J., & Sismanis, Y. (2011). Large-scale matrix factorization with distributed stochastic gradient descent. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 69-77.

Go, A., Bhayani, R., & Huang, L. (2009). Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford, 1*, 12.

Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval, 4*(2), 133-151.

Griffiths, T. L., Steyvers, M., & Tenenbaum, J. B. (2007). Topics in semantic representation. *Psychological Review, 114*(2), 211.

Griffiths, T. L., & Steyvers, M. (2004). Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America, 101 Suppl 1*, 5228-5235. doi:10.1073/pnas.0307752101

Gunawardana, A., & Shani, G. (2009). A survey of accuracy evaluation metrics of recommendation tasks. *The Journal of Machine Learning Research, 10*, 2935-2962.

Herlocker, J., Konstan, J. A., & Riedl, J. (2002). An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval, 5*(4), 287-310.

Herlocker, J. L., Konstan, J. A., Borchers, A., & Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* 230-237.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS), 22*(1), 5-53.

Herzog, D., & Wörndl, W. (2014). A travel recommender system for combining multiple travel regions to a composite trip. *Cbrecsys 2014,* 42.

Huang, Z., Chung, W., & Chen, H. (2004). A graph model for E-commerce recommender systems. *Journal of the American Society for Information Science and Technology, 55*(3), 259-274.

Huang, Z. (1997). Clustering large data sets with mixed numeric and categorical values. *Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining, (PAKDD),* 21-34.

Jamali, M., & Ester, M. (2009). Trustwalker: A random walk model for combining trust-based and item-based recommendation. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* 397-406.

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS), 20*(4), 422-446.

Jopson, B. (2011). Amazon urges California referendum on online tax. *The Financial Times*.

Kantor, P. B., Rokach, L., Ricci, F., & Shapira, B. (2011). *Recommender systems handbook*, Springer.

Karypis, G. (2001). Evaluation of item-based top-n recommendation algorithms. *Proceedings of the Tenth International Conference on Information and Knowledge Management,* 247-254.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer,* (8), 30-37.

Krulwich, B. (1997). Lifestyle finder: Intelligent user profiling using large-scale demographic data. *AI Magazine, 18*(2), 37.

Lam, S. K., & Riedl, J. (2004). Shilling recommender systems for fun and profit. *Proceedings of the 13th International Conference on World Wide Web,* 393-402.

Lin, W., Alvarez, S. A., & Ruiz, C. (2002). Efficient adaptive-support association rule mining for recommender systems. *Data Mining and Knowledge Discovery, 6*(1), 83-105.

Liu, Q., Ge, Y., Li, Z., Chen, E., & Xiong, H. (2011). Personalized travel package recommendation. *Data Mining (ICDM), 2011 IEEE 11th International Conference on,* 407-416.

Lu, E. H., Lin, C., & Tseng, V. S. (2011). Trip-mine: An efficient trip planning approach with travel time constraints. *Mobile Data Management (MDM), 2011 12th IEEE International Conference on 1*, 152-161.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1*(14), 281-297.

Manouselis, N., & Costopoulou, C. (2007). Experimental analysis of design choices in multiattribute utility collaborative filtering. *International Journal of Pattern Recognition and Artificial Intelligence, 21*(02), 311-331.

McLaughlin, M. R., & Herlocker, J. L. (2004). A collaborative filtering algorithm and evaluation metric that accurately model the user experience. *Proceedings of the 27th Annual*

International ACM SIGIR Conference on Research and Development in Information Retrieval, 329-336.

McNee, S. M., Albert, I., Cosley, D., Gopalkrishnan, P., Lam, S. K., Rashid, A. M., . . . Riedl, J. (2002). On the recommending of citations for research papers. *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work,* 116-125.

McNee, S. M., Riedl, J., & Konstan, J. A. (2006). Being accurate is not enough: How accuracy metrics have hurt recommender systems. *CHI'06 Extended Abstracts on Human Factors in Computing Systems,* 1097-1101.

Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002),* Edmonton, Canada, 187-192.

Millar, J. R., Peterson, G. L., & Mendenhall, M. J. (2009). Document clustering and visualization with latent dirichlet allocation and self-organizing maps. *Proceedings of the Twenty-Second International FLAIRS Conference,* 69-74.

Mobasher, B., Dai, H., Luo, T., & Nakagawa, M. (2001). Effective personalization based on association rule discovery from web usage data. *Proceedings of the 3rd International Workshop on Web Information and Data Management,* 9-15.

Nilashi, M., bin Ibrahim, O., & Ithnin, N. (2014). Multi-criteria collaborative filtering with high accuracy using higher order singular value decomposition and neuro-fuzzy system. *Knowledge-Based Systems, 60*, 82-101.

Nilashi, M., bin Ibrahim, O., Ithnin, N., & Sarmin, N. H. (2015). A multi-criteria collaborative filtering recommender system for the tourism domain using Expectation Maximization (EM) and PCA–ANFIS. *Electronic Commerce Research and Applications*, *14*(6), 542-562.

O'Donovan, J., & Smyth, B. (2005). Trust in recommender systems. *Proceedings of the 10th International Conference on Intelligent User Interfaces,* 167-174.

Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. (2012). A literature review and classification of recommender systems research. *Expert Systems with Applications, 39*(11), 10059-10072.

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review, 13*(5-6), 393-408.

Pazzani, M., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine Learning, 27*(3), 313-331.

Poirier, D., Fessant, F., & Tellier, I. (2010). Reducing the cold-start problem in content recommendation through opinion classification. *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), 1,* 204-207.

Powers, D. M. (2011). Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies. 2*(1), 37-63.

Pritchard, J. K., Stephens, M., & Donnelly, P. (2000). Inference of population structure using multilocus genotype data. *Genetics, 155*(2), 945-959.

Rajman, M., & Besançon, R. (1998). Text mining-knowledge extraction from unstructured textual data. *Advances in data science and classification* (pp. 473-480) Springer.

Rao, K. N. (2010). Application domain and functional classification of recommender systems—a survey. *DESIDOC Journal of Library & Information Technology, 28*(3), 17-35.

Reddy, S., & Mascia, J. (2006). Lifetrak: Music in tune with your life. *Proceedings of the 1st ACM International Workshop on Human-Centered Multimedia,* 25-34.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work,* 175-186.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An open architecture for collaborative filtering of netnews. *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work,* 175-186.

Ricci, F. (2002). Travel recommender systems. *IEEE Intelligent Systems, 17*(6), 55-57.

Ricci, F., Cavada, D., Mirzadeh, N., & Venturini, A. (2006). Case-based travel recommendations. *Destination Recommendation Systems: Behavioural Foundations and Applications,* 67-93.

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. *Proceedings of the 24th International Conference on Machine Learning,* 791-798.

Salton, G. (1992). The state of retrieval system evaluation. *Information Processing & Management, 28*(4), 441-449.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). *Application of Dimensionality Reduction in Recommender System-a Case Study (No. TR-00-043).* Minnesota University, Minneapolis Department of Computer Science.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of the 10th International Conference on World Wide Web,* 285-295.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2002). Incremental singular value decomposition algorithms for highly scalable recommender systems. *Fifth International Conference on Computer and Information Science,* 27-28.

Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative filtering recommender systems. *The adaptive web* (pp. 291-324) Springer.

Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). E-commerce recommendation applications. *Applications of data mining to electronic commerce* (pp. 115-153) Springer.

Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* 253-260.

Schumacher, M., & Rey, J. (2011). Recommender systems for dynamic packaging of tourism services. *Proceedings of the 18th international Conference on Information Technology and Travel & Tourism (ENTER 2011),* 13-23.

Shan, H., & Banerjee, A. (2010). Generalized probabilistic matrix factorizations for collaborative filtering. *2010 IEEE 10th International Conference on Data Mining (ICDM),* 1025-1030.

Shani, G., Brafman, R. I., & Heckerman, D. (2002). An MDP-based recommender system. *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence,* 453-460.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 210-217.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems,* 210-217.

Sivic, J., Russell, B. C., Efros, A. A., Zisserman, A., & Freeman, W. T. (2005). Discovering objects and their location in images. *The Tenth IEEE International Conference on Computer Vision, ICCV 2005, 1*370-377.

Smyth, B. (2007). Case-based recommendation. *The adaptive web* (pp. 342-376) Springer.

Souffriau, W., Vansteenwegen, P., Vertommen, J., Berghe, G. V., & Oudheusden, D. V. (2008). A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence, 22*(10), 964-985.

Souffriau, W., Vansteenwegen, P., Vertommen, J., Berghe, G. V., & Oudheusden, D. V. (2008). A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence, 22*(10), 964-985.

Sra, S., & Dhillon, I. S. (2005). Generalized nonnegative matrix approximations with bregman divergences. *Advances in Neural Information Processing Systems, NIPS 2005,* 283-290.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence, 2009*, 4.

Swets, J. A. (1963). Information retrieval systems. *Science (New York, N.Y.), 141*(3577), 245-250. doi:141/3577/245

Takács, G., Pilászy, I., Nemeth, B., & Tikk, D. (2008). Investigation of various matrix factorization methods for large recommender systems. *IEEE International Conference on Data Mining Workshops, 2008, ICDMW'08,* 553-562.

Takács, G., Pilászy, I., Németh, B., & Tikk, D. (2009). Scalable collaborative filtering approaches for large recommender systems. *The Journal of Machine Learning Research, 10*, 623-656.

Tan, C., Liu, Q., Chen, E., Xiong, H., & Wu, X. (2014). Object-oriented travel package recommendation. *ACM Transactions on Intelligent Systems and Technology (TIST), 5*(3), 43.

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63*(2), 411-423.

Torres, R., McNee, S. M., Abel, M., Konstan, J. A., & Riedl, J. (2004). Enhancing digital libraries with TechLens. *Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries,* 228-236.

Turney, P. D. (2002). Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics,* 417-424.

Van Rijsbergen, C. (1979). Information retrieval. Department of computer science, University of Glasgow. *URL: Citeseer.Ist.Psu.edu/vanrijsbergen79information.html.*

Vansteenwegen, P., & Van Oudheusden, D. (2007). The mobile tourist guide: An OR opportunity. *OR Insight, 20*(3), 21-27.

Vargas, S., & Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. *Proceedings of the Fifth ACM Conference on Recommender Systems,* 109-116.

Vozalis, M., & Margaritis, K. G. (2004). Collaborative filtering enhanced by demographic correlation. *Proceedings of the AIAI Symposium on Professional Practice in AI, part of the 18th World Computer Congress.*

Weng, J., Lim, E., Jiang, J., & He, Q. (2010). Twitterrank: Finding topic-sensitive influential twitterers. *Proceedings of the Third ACM International Conference on Web Search and Data Mining,* 261-270.

Willemsen, M. C., Graus, M. P., & Knijnenburg, B. P. (2014). Understanding the role of latent feature diversification on choice difficulty and satisfaction. *Under Review.*

Xie, M., Lakshmanan, L. V., & Wood, P. T. (2010). Breaking out of the box of recommendations: From items to packages. *Proceedings of the Fourth ACM Conference on Recommender Systems,* 151-158.

Yang, W., & Hwang, S. (2013). iTravel: A recommender system in mobile peer-to-peer environment. *Journal of Systems and Software, 86*(1), 12-20.

Zhang, M., & Hurley, N. (2008). Avoiding monotony: Improving the diversity of recommendation lists. *Proceedings of the 2008 ACM Conference on Recommender Systems,* 123-130.

Ziegler, C., McNee, S. M., Konstan, J. A., & Lausen, G. (2005). Improving recommendation lists through topic diversification. *Proceedings of the 14th International Conference on World Wide Web,* 22-32.