A hybrid group recommender system for travel destinations

Jeroen Dhondt

Supervisor: Prof. dr. ir. Luc Martens
Counsellors: Dr. ir. Toon De Pessemier, Kris Vanhecke

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in de ingenieurswetenschappen: computerwetenschappen

Department of Information Technology
Chairman: Prof. dr. ir. Daniël De Zutter
Faculty of Engineering and Architecture
Academic year 2014-2015
A hybrid group recommender system for travel destinations

Jeroen Dhondt

Supervisor: Prof. dr. ir. Luc Martens
Counsellors: Dr. ir. Toon De Pessemier, Kris Vanhecke

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in de ingenieurswetenschappen: computerwetenschappen

Department of Information Technology
Chairman: Prof. dr. ir. Daniël De Zutter
Faculty of Engineering and Architecture
Academic year 2014-2015
Preface

We live in a really exciting time where information technology keeps getting more important in our lives. This research was my chance to help people out in a new domain of their life: their leisure plans. By making an application that tried to understand the underlying influences for making a decision of where to travel next, and presenting the destinations that fit the user’s interests, I hope to have made the choice a little easier. These were also the reason why this topic interested me from the start.

This master thesis wouldn’t have been possible without my counselor, Toon De Pessemier, who always stayed patient and helped me where it was needed and proved a invaluable source of information and input to designing my application. Further I want to thank my supervisor, prof. Luc Martens for making available this interesting topic, and my other counselors Simon Dooms and Kris Vanhecke for their presence and feedback during all presentations.

Finally I want to thank my supporting family, in particular my father who I could always turn to for support and feedback. He was always there when I needed that extra push, and without him this thesis would have looked very different. I also want to thank my friends and other test users for taking their time to evaluate the application and give their valuable feedback. Lastly, I want to thank my girlfriend Niya for always showing interest in my work and supporting me during this time.

Jeroen Dhondt, May 2015
Permission for usage

“The author(s) gives (give) permission to make this master dissertation available for consultation and to copy parts of this master dissertation for personal use. In the case of any other use, the copyright terms have to be respected, in particular with regard to the obligation to state expressly the source when quoting results from this master dissertation.”

Jeroen Dhondt, May 2015
A hybrid group recommender system for travel destinations

by

Jeroen DHONDT

Master’s dissertation submitted in order to obtain the academic degree of
Master of Science in de ingenieurswetenschappen: computerwetenschappen

Academic year 2014-2015

Supervisor: Prof. dr. ir. Luc Martens
Counsellors: Dr. ir. Toon De Pessemier, Kris Vanhecke

Faculty of Engineering and Architecture
Universiteit Gent

Department of Information Technology
Chairman: Prof. dr. ir. Danil De Zutter

Summary

Recommender systems can help people find the items that best fits their personality and needs when they are presented with an overload of information. These systems have been used in different domains like music and movies, but should also be useful in the domain of travel. In this research, such a system is presented: the group travel recommender TravelWithFriends. The system suggests interests travel locations to a user based on their rating profile, interests and specific demands from their next destination. TravelWithFriends combines three different recomenders together to find the optimal hybrid approach for this complex domain. The system was further extended with a group aspect, allowing families and groups of friends to receive suggestions based on their combined profiles. All this was tested in a prototype web application and evaluated by a group of test users.

Keywords

Recommender system, travel, hybrid, group
A hybrid group recommender system for travel destinations

Jeroen Dhondt

Supervisor(s): Luc Martens, Toon De Pessemier, Kris Vanhecke

Abstract—Recommender systems can help people find the items that best fit their personality and needs when they are presented with an overload of information. These systems have been used in different domains like music and movies, but should also be useful in the domain of travel. In this article, such a system is presented: the group travel recommender TravelWithFriends. The system suggests interest travel locations to a user based on their rating profile, interests and specific demands from their next destination. TravelWithFriends combines three different recommenders together to find the optimal hybrid approach for this complex domain. The system was further extended with a group aspect, allowing families and groups of friends to receive suggestions based on their combined profiles. All this was tested in a prototype web application and evaluated by a group of test users.

Keywords—Recommender system, travel, hybrid, group

I. INTRODUCTION

RECOMMENDER systems have become more and more integrated with our daily (online) life. We can find them nowadays in our search engine, as personalized advertisements in our browser or helping us find interesting products in our favorite online shop. eTourism is also a booming business online, with many websites offering to help you book your vacation or find advice and information on a tourist destination.

While all these services help you plan your vacation, they skip one important question: Where do I want to go next? There is no online service that offers help with narrowing down your options to a shortlist of destinations that fit your interests. This abstract describes the research and design of such a system: the travel destination recommender TravelWithFriends.

II. RELATED WORK

A. Approaches to recommendations

In the past different approaches to designing a recommender system have been proposed[1]. While some strategies allow to directly recommend a list of items, most systems have a different approach. These systems all summarize the users and items to a representation that allows for an easy comparison. The next step is to predict the ratings of new items and present the highest scoring ones to the user. In literature, the three most common strategies have been thoroughly studied in the past: collaborative filtering, content-based filtering and knowledge-based recommendations.

In collaborative filtering, recommendations are made based on user ratings. The strategy looks for agreements in rating profile between users (or items) and presents the items that have consequently been rated high by these like-minded people. Content-based approaches look closer at the information available for the item. Keywords or tags are connected to the items, and for each user a tag profile is created. New items that most closely match the user’s profile will be recommended. Knowledge-based recommendations make a similar approach to the content-based one, but for these a more profound and domain-specific models are created to represent users and items.

B. The domain of travel

The value of recommendation systems in this industry was already seen by Ricci back in 2002 [2]. He noticed that recommending travel destinations lends itself easier to content-based and knowledge-based systems. Travel destinations can be portrayed by rather stable concepts and as such a good knowledge base can be reused by different recommender engines. However, making a fitting model to represent this domain has not yet lead to conclusive results. Liu et al. [3] also acknowledged this complexity of travel recommendations and proposed a spatio-temporal model fitting for travel packages, called the TAST (Tourist-Area-Season-Topic) model. Petrevska et al. [5] proposed a hybrid approach to recommend tourist attractions in Macedonia. They combined a content-based component base on a user profile following the tourist types described by Gibson [4] with a collaborative filter.

These related works have proved some working concepts of recommender strategies in a smaller subdomain of travel. Liu et al. focused on a catalog off fixed tours by a travel agency while Petrevska et al. started from a fixed destination, Macedonia, and suggested attractions to their users. The goal of our design is to broaden the domain to travel destinations and still make the design resistant to the common problems recommender systems face: sparse data and the cold start problem.

C. Group recommendations

Group recommenders have been researched in more recent years. Two main strategies to merge user profiles are proposed: recommendation aggregation, where individual recommendations are merged together to a final list, and profile aggregation, where the different profiles of the members are condensed into one group profile and recommendations are made based on this.

III. ARCHITECTURE OF TRAVELWITHFRIENDS

A. Individual recommendations

A new weighted hybrid approach was designed to recommend travel destinations to users. These destinations were chosen as the most popular touristic cities all over the world. The final database consisted of 685 locations, spread over all six continents. The system consist of different important building blocks.
as seen from Figure 1. The user profile consists of user inputted ratings and an interests profile, indicating the affinity a users feels towards 19 tourist profiles, like nature, adventure, food etc. Before the recommendation process starts, the user can choose his constraints to the locations: the budget he wants to spent, the attractions available at the location and the distance to travel.

The recommender engine processes the constraints to create a shortlist of destinations that can be suggested. Next, all the aforementioned information is used in three strategies. The collaborative filter predicts purely based on ratings the score for each item (how well an item matches the user profile). A second approach is the content-based filter that takes as input the tags of destinations. These tags were derived from the attractions available at the destination. A destination with a famous castle would for instance receive the tag 'Historic Site' among others. Finally a knowledge-based approach was also implemented, taking information from different resources to make a destination profile. The profile consisted of their transport cost, their affinity with each of the 19 user profiles and the activities available. After all three strategies predicted the scores for each destination, these results were merged together and the user was presented with their final recommendation.

B. Group recommendations

A second element of the TravelWithFriends application was its group recommender. This part predicted a final list of destinations best fitting a group of friends or a family in two steps (see Figure 2). First a list of ten candidates was produced by a recommendation aggregation approach. The average without misery approach was used to assure no group member would have to suffer through a vacation they absolutely hated. In a second step, the group members were asked for feedback by allowing them to rank their favorite destinations from the shortlist. Finally a simple BORDA-count decided the top locations to present.

IV. Evaluation

To evaluate the different strategies individual strengths and weaknesses, a web application was implemented in Grails, and making use of the Lenskit Toolkit to build the different recommenders. 16 test users were presented with the application and asked to answer a series of questions based on the work of Pu[6]. They were generally very pleased with the experience, noting that each recommendation brought new and interesting places to their attention, and making the application a more interactive and fun experience to explore new locations to visit.

The users were presented with five different recommendation lists (a baseline strategy was added that returned the most popular locations that fitted the constraints). To compare them, each user was asked to rank the lists and give a score in four domains: matching the profile, novelty, diversity and usefulness. The results conclusively showed that the hybrid recommender outscored the other approaches with an average rank of 2.31 and picked among the top 2 in 69% of the cases, followed by the content-based and knowledge-based (average: 2.81), the collaborative (3.25) compared to the the baseline that received an average rank of 3.81.

The answers to the questions revealed further the strengths and weaknesses of each approach. The Collaborative filter (CF) showed the highest novelty, while the content-based (CBF) and knowledge-based (KB) strategy match the user’s interests better. The evaluation of the group component revealed the 2-step approach lead to an overall high appraisal of the system and the final recommendation, where the user felt their own opinion was valued by the system. They were less certain about the inner workings of the application and asked for more transparency in the process and more explanations of the recommendations made. The user interface can be extended to take care of these concerns by the user.
V. CONCLUSIONS

In this abstract we discussed the design of a travel recommender system for groups and the prototype designed under the name TravelWithFriends. The test users were generally very pleased with the recommender, describing it as an interesting and interactive way of discovering new destinations. The research showed many different approaches can be used, each with their own merits and limitations. Bringing these strategies together into one hybrid system received the highest overall satisfaction. A design for a group recommendations was also proposed in which a recommendation aggregation step based on the average without misery strategy was combined with a feedback round where users can pick their favorites and a BORDA-count decides on the final ranking. Users were again pleased with the application’s overall performance but notices a lack of insight in the recommendation process. More ways for explaining the results should be added to the application.

REFERENCES

## Contents

1 Introduction .................................................................................................................. 1
   1.1 Motivation ................................................................................................................ 1
   1.2 Goals and outline ..................................................................................................... 2

2 Recommender systems in literature ............................................................................ 4
   2.1 Introduction to recommender systems ..................................................................... 4
     2.1.1 The anatomy of recommender systems .............................................................. 5
     2.1.2 Content-based filtering .................................................................................... 7
     2.1.3 Collaborative filtering ..................................................................................... 9
     2.1.4 Knowledge-based recommendation ................................................................... 11
     2.1.5 Hybrid recommender systems ......................................................................... 12
   2.2 Travel recommendation ........................................................................................... 12
     2.2.1 Models for travel plans .................................................................................... 13
     2.2.2 Existing systems ............................................................................................ 14
   2.3 Group recommendations ......................................................................................... 14
   2.4 Conclusions ............................................................................................................. 16

3 System overview .......................................................................................................... 17
   3.1 Application .............................................................................................................. 17
   3.2 System Architecture and Flows .............................................................................. 19
   3.3 Recommendation engine ....................................................................................... 20
   3.4 Group Recommendation ....................................................................................... 20
   3.5 Data sources ........................................................................................................... 21

4 Recommendations in TravelWithFriends .................................................................... 23
   4.1 Collaborative recommender ................................................................................... 23
CONTENTS

4.1.1 User ratings ................................................. 24
4.1.2 Neighbor selection algorithm ............................... 24
4.1.3 Item-based scoring algorithm ............................... 26
4.2 Content-Based recommender .................................... 28
  4.2.1 Destination’s tag profile ................................. 28
  4.2.2 The adjusted TF-IDF algorithm ........................... 29
4.3 Knowledge-Based recommender ................................ 30
  4.3.1 Travel destination profile ................................. 30
  4.3.2 User profile .............................................. 31
  4.3.3 Scoring algorithm .......................................... 31
4.4 Constraints and pre-filtering .................................. 32
4.5 Hybrid recommendations ....................................... 33
4.6 Conclusions .................................................... 33

5 Recommendations for groups ................................. 34
  5.1 Introduction to the group recommender ....................... 34
  5.2 Step 1: Selecting the candidates list ....................... 36
  5.3 Step 2: Final list and ranking ............................... 37
  5.4 Conclusions .................................................... 38

6 Information resources ......................................... 39
  6.1 The travel destinations ...................................... 39
  6.2 User rating dataset and tourist profiles .................... 41
  6.3 Travel costs ................................................... 42

7 System design and implementation ............................ 45
  7.1 Client-server architecture .................................... 45
  7.2 The recommender engine ...................................... 46
    7.2.1 Item-Item recommender .................................. 47
    7.2.2 Content-based recommender .............................. 48
    7.2.3 Knowledge-based recommender ........................... 49
    7.2.4 The RecommenderManager class ........................ 50
  7.3 The Grails web application ................................... 51
    7.3.1 The Domain Classes ..................................... 51
7.3.2 The Controllers ........................................... 52
7.3.3 The Services ........................................ 53
7.3.4 The Views ........................................ 53
7.4 User interface ........................................... 53
  7.4.1 Finding destinations and adding ratings ................. 53
  7.4.2 Individual recommendation ............................ 55
  7.4.3 Group recommendation ................................. 56
7.5 Data model and storage ..................................... 57

8 Evaluation ..................................................... 59
  8.1 Test setup and questionnaire ................................. 59
  8.2 General evaluation of the individual recommender .......... 60
  8.3 Evaluation of the different recommender approaches ........ 61
  8.4 Evaluation of the group recommender ..................... 64
  8.5 Conclusion ............................................. 65

9 Conclusion ................................................... 67
  9.1 Conclusion ............................................... 67
  9.2 Future Work ............................................ 68

A Evaluation results and questionnaire ................................ 70
  A.1 Evaluation results ...................................... 70
  A.2 Student’s T-test for mean rankings ....................... 73
  A.3 Questionnaire .......................................... 75
Chapter 1

Introduction

1.1 Motivation

Increasing amounts of information on traveling can be found around the world wide web. As is also the case in many other domains, the web has become the place to find information for your next holiday. Websites like SkyScanner or Expedia allow you to find the best deals, flight tickets or travel packages. Others like WikiVoyage and Frommers are specialized in giving you information and travel advice on different destinations. There’s also the ever increasing popularity of TripAdvisor, a website that allows users to rate and comment hotels, restaurants and attractions they have visited.

While all these services help you plan your vacation, from transport to accommodation and places to visit, they skip one important question: Where do I want to go next? Surely you can just go through the list of destinations or ask your friends for advice. But there is no online service that offers help narrowing down your options to a personalized shortlist of destinations that fit your interests. This master thesis has the goal to solve exactly that question, by designing a travel destination recommender.

Recommender systems are already successfully designed to help a user find personalized information in various domains like movies, music and buying products. However, there has yet to be designed a successful recommender system for travel destinations. Different reasons make this
domain a lot harder to make recommendations. Travel experiences are much more complex than the experience of watching a movie or buying a certain product. And data is harder to come by, as we experience less holidays than we watch movies or listen to music.

The research first focuses on recommending the perfect destination to a single user. But most travel plans are not made alone. Therefore the travel recommender was then extended to also allow for groups to get personalized recommendations.

### 1.2 Goals and outline

This thesis starts in Chapter 2 with a general overview of recommender systems, explaining general terminology and concepts in this domain. In this literature study you will find a selection of analyzed and compared opinions of researchers in the domain of travel and group recommenders in order to get indications on the how to build the prototype, to identify the challenges and finally get directions of how to tackle them in this recommender environment.

Chapter 3 gives a quick overview of the algorithms and design choices of the web application that will be build in the following chapters. Some simple diagrams describe the usage flow and familiarize you with the different building blocks of the application.

The goal in this research was mainly to find which model could best describe travel destinations. The different approaches and algorithms designed are presented in Chapter 4 where we show how a collaborative, a content-based and a knowledge-based approach each can be used to make a travel recommender.

Different components of group recommender systems have been analyzed to come to a recommended architecture for TravelWithFriends. Chapter 5 presents the design of our group travel recommender. This recommender is designed to take into account all the different user profiles and allows for groups to vote together where their next vacation should be.

Finding the right data resources for the domain of travel and bringing them all together also proved quite the challenge. Chapter 6 sheds some light on the different resources, the gathering process and the final information made available.
The next challenge was to build the **prototype application**. A web technology, recommender library and a database structure had to be selected. We chose the Grails web framework and the Lenskit Recommender Toolkit to build our application. The motivations and implementation of these can be found in Chapter 7.

For **evaluation** our assumptions and design, the TravelWithFriends application was tested by various users. The results and conclusions of this evaluation are presented in Chapter 8 together with the valuable feedback and lessons learned from the prototype test.

This work is concluded in Chapter 9 where first the major findings are presented, followed by ideas for further extensions of the model and future work.
Chapter 2

Recommender systems in literature

In this chapter we give an overview of previous research and existing systems in the field of recommender systems. We start off in Section 2.1 with an introduction to what recommender systems are and how they are generally implemented. The different recommendation approaches are discussed in detail, as we will use them later on for TravelWithFriends. Next, previous work in the field of travel recommendations is discussed in Section 2.2. Section 2.3 will then talk about group recommenders, their different use cases and the new challenges they bring. Finally, Section 2.4 summarizes what we have learned from existing research and how these influence the TravelBuddy recommender system.

2.1 Introduction to recommender systems

Recommender systems are made to help you in your search for a fitting product from an overwhelming array of options. Such a system can pick those items that you will enjoy most to help you in your search. Recommender systems can nowadays be found in a broad range of applications and are very common in e-business solutions. Netflix[31], an on-demand video provider, recommends products for a user in a personalized list, based on their viewing history and ratings of movies and TV series. Google’s Play Store will recommend you applications based on those you have installed on your device already.
In both cases, the recommender engine makes use of advanced algorithms to find \textbf{personalized} recommendations. Other systems make recommenders based on global trends or common-sense human behavior. Online bookshop Amazon\cite{2} for instance has a system of subtle recommendations in the form of ‘Customers Who Bought This Item Also Bought’ or by giving a selection of similar products to the one you are looking at. These \textbf{non-personalized} recommendations have their merits in certain domains but will not be discussed further in this work. In the following chapters we will focus on personalized recommendations.

\subsection*{2.1.1 The anatomy of recommender systems}

Every recommender system needs to at least consist of two base elements: the \textit{user profile} and the \textit{information filtering technique}. The \textbf{user profile} is needed for the system to represent the user’s information and preferences. Without a user profile, it becomes impossible to generate personalized recommendations. Based on this user profile, the recommender will need a certain \textbf{matching (or filtering) approach} to match users with items. In the following sections we discuss each part in more detail.

\textbf{User profiles}

A user profile contains all personal information needed of a user to help with making recommendations. Montaner\cite{30} is one of the first to analyse the different approaches to constructing and maintaining user profiles, as there are many different ways to represent the user’s preferences. Two of the most successfully used techniques are to save a user-item matrix with \textbf{ratings} a user made in the past combined with the use of a \textbf{feature vector}, representing the affinity of a user to predefined features. E.g. in a movie recommender, features can represent the genre, movie director or release year. The user profile would then represent how much the user likes each genre. The user can also be asked for \textbf{explicit input}. In such case the user can state his preferences by answering questions or indicating his interests from a list.

Recommender systems always try to improve their user’s profiles and adapt to changing user’s preferences over time. So \textit{user feedback} is another important aspect of any system. \textbf{Explicit}
feedback can be attained by asking the user to rate items or ask his opinion (like/dislike) on a recommendation. Explicit feedback is the most accurate information but ask the user for to make an effort for the system. The more user-friendly approach is to collect implicit feedback from a user’s behaviour and (natural) interactions with the system. Processing this information can also give insight to the user’s preferences.

Rating estimations and recommendations

As mentioned before, an important element in recommender systems are the user-item ratings. Ratings in recommender systems represent how pleasing or useful a certain item is to a user. When a user has experienced the product he can give it an explicit rating. But for most products, such rating is not known. Most recommender approaches reduce the problem of making a recommendation to estimating ratings for items a user hasn’t rated yet [1]. Given these estimations, the system can then recommend the highest scoring items to the user.

Filtering approaches

With the user profile and a database of items available, the final step to make a recommendation is to match users with suitable items. The filtering method decides how these are found. A broad introduction to the field of recommender systems is the book 'Recommender Systems: an introduction' (2011) written by Zanker et al. [50]. This work categorizes recommender systems by their filtering approach and distinguishes between four different ones:

- **Content-Based Filtering**, where the system makes use of the user’s profile to recommend items that exhibit similar characteristics to what he has liked in the past.

- In **Collaborative Filtering**, the recommender compares the user’s past ratings with those of other users to find users with similar taste. Highly rated items by these neighbors will be recommended.

- **Knowledge-Based recommenders** make use of domain specific information to match user interests with items.
• Hybrid systems finally, represent any system that combines two or more of the above approaches to a more complex whole.

Because of the importance of these filtering approaches to the final recommender system, a separate section will discuss each of them in detail. Content-based filtering is discussed in Section 2.1.2, collaborative filtering in Section 2.1.3, Knowledge-based in Section 2.1.4 and finally hybrid systems in Section 2.1.5.

2.1.2 Content-based filtering

Content-based recommendation methods will look at the user profile and item features to find good recommendations. The system recommends those items that exhibit similar characteristics to the user’s preferences in the past [41]. To give an example we look at the recommendation of music: when a user has in the past enjoyed listening to The Beatles and Led Zeppelin, the recommender tries to understand the common elements of these preferences. It might find a common genre (Classic Rock) and similar era (1960-1980). Based on these, items with the same characteristics are recommended: in our example, music by Jimi Hendrix and The Doors for example.

TF-IDF

A popular way to describe items for content-based filtering was adapted from the field of information retrieval. TF-IDF (Term Frequency-Inverse Document Frequency) [27] describes the item profile based on so-called terms. These can be keywords extracted from a document, or more common in recommender systems, they are (user) tags applied to the item. The item is then described by their TF-IDF vector, which gives a value to each key term by multiplying the term occurences in the document (term frequency \( f_{t,d} \)) by the inverse of the percentage of documents \( d \) this term appears in (the document frequency) to the total of documents \( D \):

\[
tfidf(t, d, D) = tf(t, d) \times idf(t, D) = f_{t,d} \times \log \frac{N}{|\{d \in D : t \in d\}|}
\]
2.1 Introduction to recommender systems

Measuring user-item similarity

To compare the user with the items, a user tag profile is needed too. This tag profile is constructed based on the user’s ratings: the tags of items the user rated highly will be present in the user profile. A common measure for matching user profiles with items is the cosine similarity [49, 1]. The similarity is based on two feature vectors: the user’s \( U \) preferences, stored in vector \( \vec{u} \), and the item \( I \)’s features stored in vector \( \vec{v} \):

\[
sim(U, I) = \cos(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \|\vec{v}\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2 \sum_{i=1}^{n} v_i^2}}
\]

Finally the items with the highest similarity can be recommended back to the user.

Problems and limitations

There are clear limitations to content-based recommender systems. Firstly, the technique is heavily based on good features available for all content. This is however hard to retrieve and in many cases the found information isn’t ideal. A possible option to get these information is automatic feature extraction from a written description. The alternative is manually entering the information. If done by field experts, this can be very time intensive. User-made information on the other hand can be imprecise and of low quality[36].

Pure content-based recommendations also suffer from a lack of quality measure. Two items with the same features are equally regarded. No distinction is made between a good quality or bad quality item.

Another problem with content-based filtering methods is over-specialization: the system will only recommend highly similar items to what the user has been interested in before. This method lacks the possibility to surprise the user and give a more unique suggestion. As [1] mentions: “Ideally, the user should be presented with a range of options and not with a homogeneous set of alternatives”. This can be resolved by inserting a certain randomness [44] to the algorithm or by removing too similar items [52].
2.1.3 Collaborative filtering

While content-based approaches will look to recommend items that show similarities to what you have liked in the past, collaborative methods will try to find similarities with other users. Two common approaches are user-based and item-based recommendation [10].

In **user-based collaborative filtering**, the system will look for other users with a very similar rating pattern for items. The system compares the user’s rating behaviour with those of others. When two users show great agreement in rating patterns, they are considered to have a similar taste. The recommender can select a group of most similar users, the so-called **nearest-neighbors**. Items that are highly rated by those neighbors but haven’t been seen by the user will be recommended.

**Item-based collaborative filtering** on the other hand, will look to find items that always receive very similar ratings from the same users. If a user rates item $A$ highly, then he always rates $B$ high too. Yet, if another user disliked item $A$, he will also give a bad rating to item $B$. Items $A$ and $B$ show similar rating patterns, and are thus considered neighbors. An item will now be recommended to the user if the user rated the other (similar) item highly.

**Measuring user-user and item-item similarity**

A good measure is needed to find these neighbouring users (or items). In content-based filtering the **cosine similarity**, was introduced (2.1.2). This proved to be a good measure to compare feature vectors. User ratings however, show a certain bias: some users always rate items highly, while some more critical users might only give a high rating to their top favorites. Cosine similarity doesn’t take this in consideration. The **Pearson Correlation Coefficient** is more suited here. In the formula for user-user similarity, $r_{u,i}$ describes the rating user $u$ gave to item $i$, while $\bar{r}_u$ is the average rating the user gave:

$$Sim(U,V) = Pearson(u,v) = \frac{\sum_{i \in I} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in I} (r_{v,i} - \bar{r}_v)^2}}$$
2.1 Introduction to recommender systems

As seen from the formula, the Pearson's correlation coefficient compares how the rating \( offset \) from the user's average rating \( \bar{r}_u \) compare. The similarity measure for item-item similarity is analogous to the above one. This measure has already proved to be successful in travel recommenders [37, 23].

Advantages over content-based

A few advantages over content-based filtering are obvious. There is no longer need for a qualitative feature descriptions of the items. Items are now solely recommended by the experience and rating of the community. This also directly implies a form of quality control: items that match the user's interests, but of low quality, will still receive low ratings. As such they are unlikely to be recommended.

Collaborative filtering methods have the advantage of inserting novelty and uniqueness (also called serendipity [15]) in the recommended items, leading to the discovery of unexpected, yet interesting items. Where content-based recommendations were bound to your previous interactions and interests, recommendations based on other users have no such boundaries. Another user that was found to show a very similar taste in one genre of items, can very well have enjoyed and highly rated a very different genre of item too.

Limitations and solutions

There are three main problems collaborative approaches suffer from:

- **Sparsity**: in typical systems the number of items available is huge and can range from a few hundred to ten-thousands or more. As such, most users will only have ratings for a very small percentage of items. This makes it hard to identify good neighbours in the system.

- **New user problem**: when a new user enters the system, he has no or very few rated items. This makes it very hard to find similar users and makes recommendations inaccurate and one-sided.
- **New item problem**: for items that have not been rated by any users, it will not be possible to calculate a rating. These new items won’t be recommended to users and as such never become a part of the system.

The sparsity problem can be solved by using additional available data, for instance from a user profile or demographics [35]. Another successful approach is the use of dimensional reduction technique like *singular value decomposition*.

The two latter problems, which are also called the **Cold Start problem** can be solved by making use of some of the advantages of content-based filtering. For this reason more and more research is done about *hybrid systems*, which will be discuss further in section 2.1.5. Another option is asking for explicit input from the user but is not seen as a recommended approach because of the reluctance of most users to fill in too much information or answers questions before they can use the recommender.

### 2.1.4 Knowledge-based recommendation

Knowledge-based recommenders will try to characterize users and items based on more domain-specific information. This information is usually provided by domain experts or inferred from different available attributes. The user can be asked explicitly to complete his profile by providing the domain information needed. An example of a domain where this is the preferred approach is real estate or cars.

When buying a new house the user will have specific demands that cannot be simply reduced to tags like the content-based approach describes. The price, size, number of rooms and location will be the most important factors in deciding. One approach in knowledge-based recommender is to make use of **Multi-Attribute Utility Theory (MAUT)** [13]. With this a (utility) scoring function can be calculated between each item and the user profile. This function takes into account the different constraints and demands by the user and matches them to the item catalog [8].
2.1.5 Hybrid recommender systems

Several hybrid approaches that combine both elements from collaborative and content-based approaches have been suggested in literature. These systems can avoid the limitations of either approach and improve the recommendations further. Various approaches to combining different algorithms are possible. For instance, Robin Burke’s article Hybrid Recommender Systems: Survey and Experiments[6] takes a look at the different possibilities.

A possible approach is to let independent recommender engines work separate and produce different sets of recommendations. Next, their results are combined and made into a final recommendation [7]. Another option is a switching strategy, where decision is made for each case and situation to which set of recommendation is used. For instance one could use a content-based approach for new users in order to overcome the cold start problem. Once a user has provided sufficient ratings, the recommender switches to a collaborative filtering approach. In [22] such a system is described, where four different recommenders are given dynamic weights to optimize to the final result.

2.2 Travel recommendation

The domain of travelling is very appealing for implementing recommendation systems [5]. The commerce value of the tourism is huge and eTourism is already very integrated with the online world. More and more people will look for travel information and inspiration online. On top of that, planning travels is usually something people take their time for, considering their different options and asking for advice from others before deciding.

The value of recommendation systems in this industry was already seen by Ricci back in 2002 [40]. He noticed that recommending travel destinations lends itself easier to content-based and knowledge-based systems. Travel destinations can be portrayed by rather stable concepts and as such a good knowledge base can be reused by different recommender engines.

A pure collaborative filtering approach is much less suited for this field because it is hard to compare two ratings for the same destination. Taking Barcelona as example, a young backpacker
can enjoy the city for its great nightlife and food. While a more seasoned couple might rate this destination high as well, but for totally different reasons: the great architecture, culture or musea present. Even the time of year you make your visit, and the context (group, budget, time spent) can make a huge impact on the final rating. A hybrid recommendation system that does take these information in consideration is much more suitable.

2.2.1 Models for travel plans

In [42] Ricci et al. proposed a Case-Based Reasoning (CBR) approach where users were presented recommendations based on previously made travel plans that are stored in the system. The system then allowed to alter and tweak these existing travel plans and saved these in the system afterwards.

Liu et al.[26] also acknowledged the complexity of travel recommendation. They noticed that travel packages have a complex spatio-temporal dimension: a trip consists of multiple geographical locations but connected to each other, and the experience is highly related to the time of year (season). Further challenges found is the limited availability of rating and the even greater sparsity of these rating compared to other domains. Users generally have experienced and can rate a lot more items in the domain of music or movies, compared to travel destinations [26].

To address these challenges, a Tourist-Area-Season-Topic (TAST) model was proposed to represent the content of travel packages. In this model, travel packages and tourists are represented by different topic distributions, where the topics are linked to the intrinsic features of the destinations visited. Their recommendation approach was a mix of content-based (mapping all users and items to the TAST-model) and collaborative filtering (finding similar minded users for the correct target season and area). In later research this model was extended with a Relation-factor to a TRAST-model [25, 16], to also represent the relationship between travellers, making this model suitable for group recommendations.

Petrevska et al. proposed a hybrid approach in [37] to recommend tourist attractions in Macedonia. For their content-based component, a user profile was constructed explicitly based on the tourist types described by Gibson [18]. This user profile was represented by a vector and further updated by feedback when a user gave ratings to attractions. The cosine similarity was used to
find the best matches between the user profile and the tourist attraction features. The second part was a cloud model collaborative filtering approach based on Wang et al. [47].

2.2.2 Existing systems

The most used resource for finding travel advice is the website TripAdvisor[46]. TripAdvisor allows users to comment and rating their different holiday experiences. They can rate the accommodation they stayed at, the restaurants they ate and the attractions they visited. TripAdvisor is a great resource to find the best place to stay, eat or spend the day once you know your destination. The website has a small recommender aspect to it in their 'Find inspiration' page where you can choose out of 9 fixed profiles and receive destinations matching the profile you chose.

Frommer’s[14] is another popular resource for information on travel destinations. They feature a list of guides and trip ideas. The website also has a slightly more advanced recommender component in their 'Dream Trip Recommender'[11] where users can choose from 18 activities or descriptions that must match their recommendation. They user can also specify the importance of four aspects of their trip: Food & Drink, Culture, Nightlife and Luxury. Based on these aspects and descriptions, a list of destinations is presented.

2.3 Group recommendations

Early on in the development of recommender systems designers of these systems already realized that the items recommended are not necessarily consumed alone. Music can be selected for a party group, movies can be seen with some friends and also most travel plans are made with others. Extending the recommender systems for individual users to groups has proven a challenging task.

A. Jameson described in [21] some of the main challenges for group recommenders and how these were dealt with in the past. The biggest challenge for a group recommender is a method
for aggregating the different user models or preferences. Two main approaches have been used in the past: profile aggregation and recommendation aggregation.

Profile aggregation techniques try to summarize different users into one profile for the group, by aggregating ratings and preferences of the individual users. The Travel Decision Forum system presented in [21] allowed users to discuss together what was the ideal destination profile for their group. Based on agreed upon group profile, a final recommendation was made.

The recommendation aggregation approach on the other hand, consists of making recommendations for each individual user and then combining these recommendations to one final list. This approach was used for instance in PolyLens[34] and Intrigue[4]. Many different aggregation strategies exist. Masthoff describes and evaluates 11 different ones for ratings in [29]. For brevity we present the four most interesting to our work here:

- **Average**: take the average rating over all group members
- **Multiplicative**: multiply the individual ratings
- **Least Misery**: take the minimum of all individual ratings as the group rating
- **Average without Misery**: take the averages over all individuals, but exclude items with individual scores that are below a certain threshold
- **Borda Count**: take the ranking of each item per user. An item scores points based on their rank: the bottom item receives 0 points, the next gets one point, etc. The points are then summed over all users.

For instance, PolyLens used the ‘Least Misery’ strategy for their recommendations of movies, making sure nobody in the group had to watch something they really hated. In Intrigue made use of the average. An experiment[28] on movie clips by Masthoff showed that multiplicative slightly outperformed the other techniques, closely followed by the average without misery strategy.
2.4 Conclusions

This chapter has shown many different approaches exist to making recommendations. While there has been some research to recommending travel destinations, no ideal solution has been brought forward. The travel recommender we design will need to cope with the challenges that other recommender systems also faced, like the cold start problem and dealing with sparse data. Another challenge comes forth of the complexity of the travel domain. A complex model will be needed that can encompass the different aspects of travel destinations.

Group recommenders have proven an interesting extension to the recommender approaches for individual users. They come with an array of new challenges and design choices. The most important one is to decide on the most fitting aggregation method for the recommendation domain.

Both for individual and group recommendations in the travel domain, there have been some great ideas have been proposed in the past, but there’s has not yet been any comparison to how these strategies compare and if some of these approaches can be combined to a more precise system. TravelWithFriends will try to deal with these challenges and combine different recommendation approaches to a fitting hybrid recommender for travel destinations.
Chapter 3

System overview

This chapter will introduce the TravelWithFriends web application. Here, the different building blocks of the recommender will be presented. Each of these building blocks will then be discussed in more detail in one of the following chapters. This chapter serves only as a short introduction.

To start off, Section 3.1 describes the general idea of the TravelWithFriends application, its main building blocks and some of its design choices made. The different user flows traversed in the recommendation process are described further in Section 3.2. Section 3.3 describes the recommender engine, the heart of our application. Next the group recommender component of the system is introduced. This chapter ends with a discussion on the different data sources used in TravelWithFriends in Section 3.5.

3.1 Application

TravelWithFriends is a travel recommender system that helps users find their ideal next travel destination. The application is presented through a website, accessible in a standard web browser. Each item recommended to the user is a city, known for its touristic value. The database holds 685 famous (and less famous) tourist locations at this point.

To present recommendations to a user, the application first needs to learn more about the user
3.1 Application

Figure 3.1: Overview TravelWithFriends application

by collecting information and summarizing this in a user profile. Each user can state their previous travel experiences by giving ratings to destinations he has visited in the past. He can also indicate his interests for 19 travel categories. This information allows the recommender to construct the user profile. Next, the user is asked for his constraints for their next destination, where he can indicate his budget preferences, preferred continent and more for this particular recommendation.

To make recommendations in the complex field of travel, multiple recommender approaches are combined to a weighted hybrid recommender. The application makes use of a collaborative, a content-based and a knowledge-based recommender. The different recommenders used are discussed more in Section 3.3, and Chapter 4 is entirely devoted to the underlying recommendation algorithms that support them.

The user would finally be presented with one personalized list of recommended destinations to visit. For this research however, the user was presented with five different lists, each list based on a different algorithm. This allows for different approaches to be compared (see Chapter 8. From the list(s), the user can then choose to find out more about a destination that interests him. Or he can give feedback on the presented list or adjust his personal information and constraints to get a new, updated recommendation.
3.2 System Architecture and Flows

The information in the system follows a fixed flow before it’s presented as a final recommendation to the user. Image 3.2 gives an overview of the system as well as the main information flow (represented by the red numbers) used in the TravelWithFriends system. The three main parts of the system are the data sources (blue elements), the recommender engine and the user interface.

![Figure 3.2: Data flow and the recommender engine](image.png)

The main information flow through the system can be split up in these steps:

1. Creating the **User Query**: the user selects his interests and destination constraints

2. **Constraint pre-filtering step**: the destinations from the database are compared to the user query and a candidates shortlist is constructed. This step reduces the possible destinations to be recommended, to only contain options that interest the user.

3. The different **recommenders** of the recommender engine each give a score to all items.

4. The scores are **merged** to a final ranking.

5. The user is presented with the **final list** of recommendations.
3.3 Recommendation engine

The recommender engine is the most crucial part of the system. Its goal is to estimate user ratings and interests based on his profile. This allows the recommender to present each user with a personalized, ranked list of destinations of top destinations to go, based on his last query. TravelWithFriends combines three separate recommenders together in order to find the best recommendations.

Before any recommendations are made, all possible destinations go through the pre-filtering phase. Destinations that don’t fulfill the constraints by the user AND the destinations already rated by the user are removed from the shortlist (we don’t want to recommend the same place again!).

The shortlist is then presented to the different recommender approaches. The collaborative recommender makes use of an item-item collaborative filtering approach. This recommender looks for similar items based on all users’ rating histories. It therefore needs access to the rating history, which is stored in a dedicated table in the database (see next Section). There is also a content-based recommender present. This recommender takes descriptive information (tags) of destinations stored in a database and matches it to the specific user’s profile. Lastly, there is the knowledge-based approach that makes use of deeper connections and information provided by domain experts to make a recommendation that takes multiple factors into account. The approaches will be discussed in detail in the next Chapter (4).

Each separate recommender algorithm produces the list based on scores. These scores indicate how suitable this destination is for the user. These scores are then merged together to present a final list of recommendations to the user.

3.4 Group Recommendation

TravelWithFriends also allows users to create groups or join an existing group of friends. Recommending destinations to a group takes two steps. First, the system makes a shortlist of destinations for this group, taking into account the ratings and preferences of each individual
member of that group. Next, each group member gets the chance to rank this shortlist of possible destinations, after which the system makes a fair and balanced review and presents the final recommendation: the ranked top 5 of suggested destinations. More details on the group recommendations can be found in Chapter 5.

3.5 Data sources

Different information sources are used in the TravelWithFriends application. These resources can all be categorized in one of two main domains: the travel destination domain or the user domain.

For the domain of travel destinations, two crucial information resources are stored:

- the Travel Destination database, consists of general information of each destination. This database contains (links to) descriptions of each location, location coordinates and country and region information.
- the Domain Knowledge database, including specific domain knowledge such as a mapping of locations and tourist profiles, attraction types and average transport costs.

In the user domain, two important information sources are saved.

- the user ratings database keeps a history of ratings a user gave to destinations he visited in the past, as well as implicit feedback provided by the user that includes what places he has visited (without giving an explicit rating).
- the user profile database keeps more general information stored from a user, including user login information, his interests profile and other information that can also be relevant for recommendations.

Chapter 6 discusses the different information resources used by TravelWithFriends and how this data was collected. In Chapter 7 the implementation of the web application is presented, and
in particular Section 7.5 presents how the data was exactly represented and stored in a MySQL database.
Chapter 4

Recommendations in TravelWithFriends

Three different recommendation algorithms have been implemented in TravelWithFriends. Each approach is discussed in the following sections: collaborative filtering recommender 4.1, the content-based recommender 4.2 and the knowledge-based approach 4.3. Further on, more details are presented on the pre-filtering step 4.4 and on how the different recommendations were merged into one, hybrid recommender 4.5. This chapter ends with a final discussion and some conclusions 4.6.

4.1 Collaborative recommender

The first recommendation algorithm of TravelWithFriends is an item-based collaborative filtering (CF) approach that makes use of the mixed set of explicit ratings and implicit feedback given by users on Gogobot.com [19]. How these rating information were collected is discussed in the Chapter 6 on data sources. To make recommendations to users with this dataset requires some creativity to adjust the traditional CF algorithms, as discussed in the following sections.
4.1 Collaborative recommender

4.1.1 User ratings

TravelWithFriends makes use of a mixed data set containing two sorts of ratings for destinations. The first category is an explicit rating. Users have this explicit rating if they have rated the destination (from 1 to 5 stars) or if they have rated attractions of this destination, in which case the average of those ratings represents the destination’s rating. Alternatively, the system collected each destination a user indicated he has 'Been Here’ to the destination or an attraction in that city as implicit feedback. Our system also keeps track of this implicit feedback as they contain two important pieces of information: 1) the user has been to this place and will not find it interesting to have it recommended to him, and 2) the user has shown interest in this place, indicating at least that this destination interested him.

A simple example of some user rating profiles are depicted in Table 4.1 where implicit feedback is shown as 'B'.

<table>
<thead>
<tr>
<th></th>
<th>User t</th>
<th>User u</th>
<th>User v</th>
<th>User w</th>
<th>User x</th>
<th>User y</th>
<th>User z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item A</td>
<td>5</td>
<td>4</td>
<td>-</td>
<td>-</td>
<td>B</td>
<td>-</td>
<td>B</td>
</tr>
<tr>
<td>Item B</td>
<td>-</td>
<td>B</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Item C</td>
<td>-</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>-</td>
<td>B</td>
<td>4</td>
</tr>
<tr>
<td>Item D</td>
<td>2</td>
<td>B</td>
<td>B</td>
<td>3</td>
<td>4</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>Item E</td>
<td>-</td>
<td>3</td>
<td>B</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 4.1: User rating profile

4.1.2 Neighbor selection algorithm

The item-based collaborative approach takes two steps to predict the rating of an unseen destination for the user: first it will select a collection of most similar destinations called the neighborhood of this item. These neighbors are found by calculating a similarity measure based on the rating profiles of the destinations, of all other destination to this destination. Then the algorithm picks the most similar items as neighbors. Finally, the final predicted rating for each unrated item will be calculated based on these most similar items.
This approach is also commonly used in Machine Learning and Data Mining and coined by the abbreviation kNN (k-Nearest-Neighbors) to describe an algorithm that selects the $k$ most similar items. As mentioned, to do this we need a similarity measure based on the ratings, that can make use of both our explicit ratings and implicit feedback. Our approach is to map our ratings to a binary value:

$$R^i_{(item)} = \begin{cases} 
1, & \text{if item was rated OR item was tagged 'Have Been'.} \\
0, & \text{otherwise.} 
\end{cases}$$

The two most commonly used similarity measures were discussed before: the Pearson Correlation coefficient 2.1.3 and Cosine Similarity 2.1.2. Pearson Correlation can only be calculated with explicit ratings and can have no meaning with binary values. The cosine similarity however can be used on binary values, but will not give the wanted result in this case. This can be seen from following example, making use of the data from Table 4.2. This table converted the ratings from Table 4.1 to their binary value. We calculate the cosine similarity between items D & C and E & C:

\[
sim(D, C) = \cos(\vec{d}, \vec{c}) = \frac{\sum_{i=1}^{n} d_i c_i}{\sqrt{\sum_{i=1}^{n} d_i^2 \sum_{i=1}^{n} c_i^2}} = \frac{(1,1,1,1,1,1) \cdot (0,1,1,0,1,1)}{\sqrt{7 \cdot 5}} = 0.84
\]

\[
sim(D, E) = \cos(\vec{d}, \vec{c}) = \frac{(1,1,1,1,1,1) \cdot (0,1,1,0,0,0)}{\sqrt{3 \cdot 5}} = 0.77
\]

<table>
<thead>
<tr>
<th>User</th>
<th>User t</th>
<th>User u</th>
<th>User v</th>
<th>User w</th>
<th>User x</th>
<th>User y</th>
<th>User z</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item A</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Item B</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Item C</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Item D</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Item E</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.2: User binary rating profile

From this example we can see where the cosine similarity falls short: a very popular item like item D outscores a less popular one (item E) only because it has much more ratings by
users. The users that rated item $E$ had all also rated item $C$ (3 out of 3), which should imply a higher similarity than item $D$ where only 5 out of 7 users rated both $C$ and $D$.

Another approach for binary (implicit) data was proposed in [9] making use of the conditional probability $P(j|i) = \frac{P(i,j)}{P(i)}$. This function shows a better relation between two items, but favors destinations with lots of ratings (in our example, item D would always receive maximum similarity). Therefore [9] suggested an altered version where another term $(P(j))^\alpha$ was added to the denominator to balance this out, leading to following similarity measure used in our system:

$$sim(i,j) = \frac{P(i,j)}{P(i) * P(j)^\alpha}$$

In our recommender system, $\alpha$ was set to be $\alpha = 0, 2$, as this value proved from empirical research to give the right balance between popularity of an item and the similarity. We show the results this has on similarity between items $D \& C$ and $E \& C$:

$$sim(D, C) = \frac{P(D, C)}{P(D) * P(C)^\alpha} = \frac{5/7}{(5/7) * (7/7)^2} = 1$$

$$sim(E, C) = \frac{P(E, C)}{P(E) * P(C)^\alpha} = \frac{3/3}{(5/7) * (3/7)^2} = 1.65$$

From this example we can see that with the (adjusted) conditional probability measure, item $E$ did receive a higher similarity than item $D$!\(^1\).

### 4.1.3 Item-based scoring algorithm

With the similarity measure ready, the k-nearest neighbors can now be selected. We select $k = 20$ and denote the neighborhood of destination $i$ for user $u$ as $N_u(i)$. Note that this neighborhood consists of the most similar items $j$ that user $u$ has rated. As such, this neighborhood is different for each user.

Next we can use the weighted sum scoring function with mean centering discussed in [10]

\(^1\)Here we found a similarity > 1. In reality our data is much more sparse and values above 1 (although possible) have not been registered.
with the items that have explicit ratings, to make a rating prediction $\hat{r}^e_i$:

$$\hat{r}^e_{u,i} = \bar{r}_i + \frac{\sum_{j \in N^e_u(i)} \text{sim}(i, j)(r_{u,j} - \bar{r}_j)}{\sum_{j \in N^e_u(i)} |\text{sim}(i, j)|}$$

To also take into consideration the information of our implicit feedback, a second scoring function was designed for the binary data defined in the previous section:

$$\hat{r}^i_{u,i} = \frac{\sum_{j \in N^i_u(i)} \text{sim}(i, j) \ast \bar{r}_j}{\sum_{j \in N^i_u(i)} |\text{sim}(i, j)|}$$

Note here again, that the neighborhood $N^i_u(i)$ can very well be different from the one used for explicit ratings: here it consists of the 20 most similar items among both rated items by user $u$ (explicit) and items the user indicated he has visited (implicit).

Finally we take a weighted sum of both rating predictions as suggested also in [51], to mix implicit feedback with explicit ratings. The weights $\alpha$ and $\beta$ were set to: $\alpha = 2 \ast \#N^e_u(i)$ and $\beta = \#N^i_u(i)$.

$$\hat{r}_{u,i} = \frac{\alpha \ast \hat{r}^e_{u,i} + \beta \ast \hat{r}^i_{u,i}}{\alpha + \beta}$$

The neighborhood size $\#N^e$ and $\#N^i$ have a fixed maximum size of 20. Some items have few ratings and lack enough suitable neighbors. So $\#N$ is within the range $[0..20]$. $\alpha$ and $\beta$ were chosen so that if both neighborhoods contain the same number of items, then the rating prediction from the explicit ratings get $2/3$ of the importance versus only $1/3$ for the prediction by the implicit feedback. If however the neighborhood for the explicit ratings has much fewer items than the one for implicit feedback, then the weight is shifted more towards the prediction with the implicit feedback.

Finally, if $N^e_u(i)$ has fewer than 5 items (and this implies $N^i_u(i)$ has too) then we regard the collaborative filtering approach not precise enough. In those cases another approach, such as the content-based approach discussed next, is used as fall-back rating.
4.2 Content-Based recommender

As mentioned before in 2.1.2, content-based recommendation methods look at the describing features of a particular item and try to find matches between these features and the user’s profile. The content-based recommender in TravelWithFriends works in the same way. To describe the items, a common approach is to analyze textual descriptions of these items and extract keywords from them. This approach could also be followed in the field of travel destinations, but has been proven a hard and painful process to lead to optimal results. Therefore, in the TravelWithFriends application, another approach was taken, one that is unique to the field of travel destinations.

4.2.1 Destination’s tag profile

This approach takes advantages of the well documented information of Points-Of-Interest. The idea is that a travel location can be approximately described by the categories and keywords linked to their main attractions (or POIs). Our application makes use of the descriptive tags of TripAdvisor’s attractions, but other data sources like those mentioned in 3.5 could have been used to the same result. The tags of attractions on TripAdvisor are chosen from a fixed set of attraction categories and restricted to one (final) node.\(^2\)

Let us take a look at an example: Paris. Among its most prominent tourist attractions are the world famed musea 'Musee du Louvre’ (categorized as [Art Museum, Museums] on TripAdvisor), and 'Musee d’Orsay’ [Speciality Museum, Museums]. Paris features as well some well known landmarks such as the 'Eiffel Tower’ [Points of Interest & Landmarks, Sights & Landmarks], 'Arc De Triomphe’ [ Architectural Buildings, Historic Sites, Sights & Landmarks] and the ’Notre Dame Cathedral’ [Religious Sites, Sights & Landmarks]. These key attractions already give a good overview of what Paris has to offer for tourists.

\(^2\) We can argue that this approach is not purely content-based, as the data is extracted from the attractions linked to the destination. Besides, it makes use of domain-knowledge by the creator of the listing, who added the tag. This approach doesn’t however process the information further and as such differs from the knowledge-based approach described further. For this reason, we refer to this approach as the content-based algorithm.
4.2 Content-Based recommender

Based on this information, a content-based recommender was constructed based around the TF-IDF (Term Frequency-Inverse Document Frequency) measure. It was based largely on the most used algorithm discussed in [27] with one adjustment: the left term was changed to the square of original term frequency. The formula below shows the TF-IDF weight for tag $t$ of destination $d$, part of the collection of all destinations $D$. Here, $N$ is the number of destinations in $D$, and $f_{t,d}$ is the frequency of tag $t$ in document $d$.

$$TFIDF(t, d, D) = \sqrt{f_{t,d}} \times \log_2 \frac{N}{|\{d \in D : t \in d\}|}$$

The adjustment to the frequency term was necessary because the tag frequency was not just set to the number of attractions tagged with this tag, but rather multiplied by the number of reviews of this attraction. By multiplying the tag by the number of reviews, more popular destinations get greater importance in defining the tags for the destination. To explain this with our above example of destination Paris for instance: the tag 'Speciality Museums' (attached to Musee D’Orsay) was applied 26.149 times (the number of reviews) to Paris. On the other hand, the tags applied to the Parc des Buttes Chaumont (the #50 most popular attraction in Paris) only get a weight of 548, the number of reviews for Parc des Buttes Chaumont.

Using the pure frequency terms of the tags in the original algorithm, proved to give too much weight to the top attractions however. In very popular destinations like Paris and Barcelona,
the top 3 most popular attractions would make up for almost one third of all reviews. If any of these attractions had a rare tag (and thus a very high IDF-term), this term would dominate the recommendations. This was exactly what happened to Barcelona, where the Sagrada Familia’s rather rare tag ‘Religious Sites’ defined all received recommendations. The result was Barcelona’s most similar destinations were locations, renowned for their beautiful cathedrals, including Santiago de Compostela, Cologne and Rouen.

In [27] the logarithm of the term frequency is suggested as an alternative weight. However, this solution shifted the importance too much to less popular attractions. Finally an analysis of different weighting terms showed the square root of the term frequency to give the right balance between both popular and less popular attractions. By taking the square of the tag frequency, the weight of these top attractions was reduced to make their influence more in balance with the other attractions, but still sufficiently enough to stand out of the less important attractions.

[maybe add a table with some example data of TF, IDF and the different weights to show the influence?]

4.3 Knowledge-Based recommender

The third recommender algorithm designed in TravelWithFriends is one based on the specific domain knowledge of travel destinations. It differs from the content-based algorithm in the way that it further processes existing information and collects other information unique to this domain, whereas the previous algorithms for item ratings and tags could easily be used in other domains for recommendation too.

4.3.1 Travel destination profile

Different information sources were found (see also Chapter 6) to improve the profile of each destination. Different information sources were integrated in the TWF system to optimize recommendations:
1. geographic information: the exact location (latitude and longitude), continent and country of each destination

2. travel costs: the costs of traveling from your current location to the destination in question

3. tourist profile: how each location matches to specific tourist profiles as defined in Gogobot [19]

4. attraction types: what specific attraction types can be found at that destination

### 4.3.2 User profile

To make recommendations, we also need to map the user’s preferences in his profile. The tourist profile of each user is saved in his account and constructed in one of two possible ways: 1) by picking manual what profiles he feels match his interests the best, or 2) by summarizing his explicit ratings, similar to the way users were matched to tag profiles in the content-based approach. All other information is gathered through the user interface just before making recommendations, by asking him for his preferred characteristics for his next travel destination [see also figure next section].

### 4.3.3 Scoring algorithm

To pick the top recommendations based on this various information, a multi-input utility function was constructed to use all of this information. The function finally takes a weighted sum over all scores to make a final predicted rating:

$$\hat{r}_k = \frac{\sum_{k \in D} w_k \cdot sc(i, k)}{\sum_{k \in D} w_k}$$

With $D$ the different domain knowledge selected by the user. The user can choose to not let certain information have an influence on the recommendations. In those cases, the weight $w_k$ is set to 0 for this domain. The different scoring algorithms and weights are summarized in the following table:
4.4 Constraints and pre-filtering

To make good travel recommendations, the first thing a good travel agent will do is ask for your demands on the budget, holiday length and accommodation expectations. The TravelWithFriends system will filter out impossible destinations from the recommendation process in a similar matter. The user interface (see Figure 4.2 which allows the user to pick some characteristics of their ideal travel location and can tick off each as a ‘hard constraint’, meaning this constraint has to be met.

<table>
<thead>
<tr>
<th>Domain</th>
<th>scoring $sc(i, k)$</th>
<th>$w_k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geo Location</td>
<td>$1 - \left( \frac{\max(0, \text{distance(user, location)} - \text{max_distance})}{\text{max_distance}} \right)^2$</td>
<td>1</td>
</tr>
<tr>
<td>Travel costs</td>
<td>$1 - \frac{\max(0, \text{expected_cost} - \text{max_budget})}{\text{max_budget}}$</td>
<td>1</td>
</tr>
<tr>
<td>Tourist profile</td>
<td>cosine similarity</td>
<td>1</td>
</tr>
<tr>
<td>Attractions</td>
<td>$\frac{# \text{types matched}}{\text{total} # \text{types requested}}$</td>
<td>$1/2$</td>
</tr>
</tbody>
</table>

Figure 4.2: Choose filters and contraints page

Hard constraints are mandatory to be met by any recommendation, and any destination that does not meet one of these constraints will never be recommended. In our system, the user can set hard constraints on the budget, the travel distance and the attraction types mandatory to be at the destination. These settings can also not be set as a hard constraint, in which case TravelWithFriends uses these as input to the knowledge-based recommender and tries to fit all preferences as good as possible.
4.5 Hybrid recommendations

While all three different approaches can score items and make recommendations on their own, merging the results together takes us of all different information sources and should make up for an even better recommender. To merge the results, a simple weighted sum of all three predicted scores are taken. The different indices are \( cbf \) for content-based filtering, \( col \) for collaborative filtering and \( kb \) for knowledge based.

\[
\hat{r}_{\text{hybrid}} = (w_{cbf} \cdot \hat{r}_{cbf}) + (w_{col} \cdot \hat{r}_{col}) + (w_{kb} \cdot \hat{r}_{kb})
\]

\[
w_{cbf} + w_{col} + w_{kb} = 1
\]

These weights are not static, but influenced by different factors. When not enough neighbors are found for collaborative filtering, \( w_{col} \) is set lower (or even zero). The knowledge-based term \( w_{kb} \) gains importance when more (soft-)constraints are given by the user. The content-based algorithm always works well and is the perfect fall-back algorithm when both other approaches show little confidence.

4.6 Conclusions

The recommender designed for TravelWithFriends has many different approaches combined to one hybrid recommender. A item-based collaborative filtering approach takes both implicit feedback and explicit ratings to into account for making recommendations based on similar rating profiles of destinations. The content-based recommender takes user-applied tags of the different attractions of a destination and uses theses to summarize each location. This approach looks to recommend destinations with similar tags as the user’s highest rated destinations. Finally a knowledge-based approach makes recommendations based on the interests of the user, his budget and distance preference. All different recommenders were then brought together in a weighted hybrid recommender. The recommenders designed in this chapter are ready to be implemented and tested for their validity, which we will do in Chapters 7 and 8.
Chapter 5

Recommendations for groups

The previous chapter discussed the recommender system helping an individual user to find new destinations to visit. Very few travel plans are made alone however: \textbf{we mostly travel in groups} of friends, as a couple or with our family. Here comes the second part of the TravelWithFriends application into play: its group recommender. The \emph{group recommender} will merge the different profiles of the group together and take each member’s opinion in consideration, to come to a recommendation the system believes to be a fair and balanced compromise for the group.

In Section 5.1 the group recommender and the different steps that lead to a group receiving their recommendation is presented. Next, Sections 5.2 and 5.3 discuss in more detail the two important steps towards making the group recommendation: selecting a shortlist of ten candidates and making a final ranking. Finally in Section 5.4 some conclusions and remarks on our approach are made.

5.1 Introduction to the group recommender

When designing a recommender system for groups, a few new elements have to be taken into account that were not an issue with individuals. The first important design decision to make is how to \textbf{merge the different opinions} and preferences of the different group members. The
5.1 Introduction to the group recommender

choice was made for a recommendation aggregation technique (as discussed in Section 5.2). With this approach a recommendation list of ten candidate destinations can then be selected.

Contrary to individual recommendations, where a user can decide for himself to skip certain recommended items and pick his favorite destination from the list presented to him, group recommendations need more communication among its members. There is the need for individual users to communicate their concerns and preferences after the first recommender step. A second step where users can explicitly rank the candidates, allows for this much needed feedback (see Section 5.3).

Bringing these design decisions together lead to the following flow of the group recommendation process:

1. Users complete their individual profiles and join a group
2. One group member starts the recommendation by selecting the constraints for the group
3. The recommender engine calculates the scores for each item for all users and combines them
4. A list of ten candidates is chosen and presented to the group (\(= \text{Step 1}\))
5. Each group member gives a personal ranking to these candidates
6. The group receives their final top 5 list (\(= \text{Step 2}\)) based on a BORDA count

![Figure 5.1: Different steps in the group recommender process](image-url)
Figure 5.1 shows this process visually. The two steps executed by the recommendation engine are presented in more detail next.

## 5.2 Step 1: Selecting the candidates list

Two different approaches to have been suggested in research (see 2.3) to get a recommendation for a group of users: profile aggregation and recommender aggregations. In TravelWithFriends, the choice was made for a recommendation aggregation technique. The profile of each individual user is first used to score all destinations. These scores are then averaged for the group to lead to a final score for each destination. The recommendation aggregation method allows us to use the average without misery strategy whereas this is harder to achieve with a profile aggregation method.

While taking the average of all scores makes for the most balanced recommendation, it suffers from a clear disadvantage: when one user has a very poor opinion on one destination that scores high for all other users, this destination might still get recommended. This situation we want to avoid in a travel recommender, as we do not want one group member to dread going on this holiday just because the destination really pleases the other group members. We implement an average without misery approach to alleviate this problem: if a destination scores really low for one user, the destination is removed from the possible candidates. Here, we have set the threshold to a score below 50% of the highest scoring destination for that user. A short example will clarify this (see Table 5.1).

<table>
<thead>
<tr>
<th></th>
<th>Destination A</th>
<th>Destination B</th>
<th>Destination C</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>0.2</td>
<td>0.6</td>
<td>0.6</td>
</tr>
<tr>
<td>User 2</td>
<td>0.9</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>User 3</td>
<td>0.9</td>
<td>0.5</td>
<td>0.3</td>
</tr>
<tr>
<td>Average</td>
<td>0.67</td>
<td>0.57</td>
<td>0.47</td>
</tr>
</tbody>
</table>

Table 5.1: Group scoring example

In the example, we can see three users with their scores for three destinations. Destination A
5.3 Step 2: Final list and ranking

From the first step we have come up with a list of ten candidates for our final recommendation based on the user profiles. The second step allows for the group members to give feedback on the list and each choose their favorites. In TravelWithFriends, the user is invited to rank each destination on the list (each value from 1 to 10 can be chosen once). Any destinations without ranking are automatically set equal to rank 10. Next a BORDA count is used to pick the final order. The BORDA count is the sum of the rankings given by the users. The destination with the lowest BORDA count get picked first. The group is finally presented their top 5 destinations. Table 5.2 shows an example. In this situation, the group would be presented with final recommendation: D(7)-B(8)-H(10)-I(15)-F(18).

\[\text{Table 5.2: Group ranking example with BORDA count}\]

<table>
<thead>
<tr>
<th>Destination</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 1</td>
<td>4</td>
<td>1</td>
<td>-</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>-</td>
<td>3</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>User 2</td>
<td>-</td>
<td>3</td>
<td>-</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>User 3</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>-</td>
<td>7</td>
<td>2</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>BORDA</td>
<td>23</td>
<td>8</td>
<td>23</td>
<td>7</td>
<td>19</td>
<td>18</td>
<td>23</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
</tbody>
</table>

is the clear winner when we select by average score. However, this destination only scores 0.2 by User 1, proving to be a very poor choice for him. Our average without misery removes this option because it scores lower than the 50% threshold (0.2 < 0.6 * 50%). For the same reason Destination C will not be accepted as an recommendation because it scores below the threshold for user 3 (0.3 < 0.9 * 50%).

After scoring all locations, any destination rated by one of the users is taken out of the list of destinations that can be recommended. We assume all members prefer to visit a place they haven’t seen before. Finally, the list of ten candidates is picked as the ten highest scoring destinations from the remaining list.

---

1 Notice the slight difference with the BORDA count discussed in Chapter 2 where the last destination scored 0 and the first got the maximum score. Here we simply take the lowest score as winner, leading to the same result.
5.4 Conclusions

In this chapter we have presented the group recommender used in TravelWithFriends. The approach first merges the different individual recommendations to come up with a list of top ten candidates for the group recommendation. A second step allows the users to pick their favorites by ranking the list and based on these rankings a top 5 is deduced by taking the BORDA count.

A common point of discussion for group recommenders is their susceptibility to manipulation by users with bad intentions. In the context of travel groups, this element should not be a large concern, as the users at least like each other enough to go on a holiday together. Still, a user could try to get his favorite destination on the top of the list. In TravelWithFriends this is countered in two ways: 1) by removing the elements causing 'misery' in the first step, where the user cannot manipulate the results and 2) by forcing each user to at least rank his top 5 destinations. If he was allowed to just pick his favorite one (and leave all other to rank 10), his personal favorite would gain too much weight.

Finally some extensions to our model could be thought of to further improve the group recommender. It is yet to be proved that the hybrid recommender approach is the most suitable for merging user preferences. Instead, one could for instance use the content-based, knowledge-based or collaborative approach of the individual recommender. The recommendation aggregation could also be swapped out or combined with a profile aggregation approach: the user ratings, interests and set constraints could be merged to one profile, and a top ten recommendation could then be searched for this profile. A last point of improvement would be to have more communication between group members, allowing them to explain their ranking and allow more internal deliberation.
Chapter 6

Information resources

Making a database for a travel recommender system has proved quite a challenge, as no such datasets are publicly available. The data used in TravelWithFriends has been collected through different database dump and by scraping of information from various websites. This chapter reveals some of the information sources used in the application. Section 6.1 explains how the list of destinations was formed. The following sections 6.2 and 6.3 tell more on how the data set and transportation costs were collected respectively.

6.1 The travel destinations

There are plenty of open-source data projects that include POI’s (Points-Of-Interest) like Google Places, Yelp or Yahoo Local. These projects contain lots of useful data, but lack information on tourist attractions or the distinction between regular cities and those places interesting to tourists. Other data sources with similar information are the ontology knowledge bases like YAGO and Freebase. However, at this point in time, these didn’t have complete data on travel destinations, and are therefor not a suitable option.

As starting point for the database of our system, the free available data dump of WikiVoyage[48], the Wikipedia alternative for travel destinations, was used. This data is available under the license by the WikiMedia Foundation, which also encompasses Wikipedia. Their data
6.1 The travel destinations

consisted of over **26000 locations and tourist information pages**, created by users. This data has the big advantage that all entries have specific touristic value, and they come with a description and information useful for tourists.

![Image of WikiVoyage page on London](https://en.wikivoyage.org/wiki/London)

Figure 6.1: WikiVoyage page on London (https://en.wikivoyage.org/wiki/London)

However, lots of these pages were not actual destinations but rather collections of locations, information on a specific tour or travel tour etc. To select only the actual locations, the data was connected to database of GeoNames [17], a database listing over 100,000 place names in the world with their **geographical data**. This brought the number of listings down to 6900 actual cities and villages.

Still, many of these locations were obscure, small locations that may be interesting to explore while in the neighborhood, but that cannot be a sufficient reason to recommend these places as the main destination for a travel plan. As such, another filter was needed to only recommend 'sufficiently relevant' places. This filter used the popularity (in the form of number of ratings given) on the popular travel reviews website TripAdvisor [46]. The threshold for being considered a sufficient 'relevant' location was set to having at least 25,000 reviews on TripAdvisor. In the end, a database of **685 top locations** was formed.
6.2 User rating dataset and tourist profiles

Another important source for data was the website Gogobot [19], a travel application website that lets their users rate travel destinations and attractions. The very interesting part of Gogobot over similar social travel networks like TripAdvisor is their use of tribes. Tribes represent different tourist profiles which allows user to identify them with one or more of these 19 groups, and receive personalized feedback. Image 6.2 shows some of these tribes and how they are portrayed on the website of Gogobot.

![Gogobot Tribes](http://www.gogobot.com/tribes)

Two sources of data were collected from Gogobot.com. Firstly, the website was a source for user ratings. As discussed in the section explaining the collaborative recommender approach, this data consisted of explicit ratings (when the user had given 1 to 5 stars to a particular place or attraction) and the implicit feedback of 'Have Been', indicating that the user has visited this location.

The second source of reviews were the destination pages themselves. But this time, tribe information was collected. Users could give tribe-specific information on the destination in two ways. The first method is by giving an explicit 'Recommended for' recommendation. The second way was as an implicit feedback when they had indicated in their profile which tribes best match their interest, and then rated this destination. The explicit feedback was given twice the amount
of weight in the database, and surpasses the implicit information. Image 6.3 shows an example, where a user has indicated affinity with 7 different tribes, including business travelers (the little tie icon) and green travelers (the leave icon). In this review however, the specific recommendation for the 2 tribes 'Local Culture' and 'Foodies' surpasses those memberships and only the explicit recommendations are remembered.

![Image of a user review](image)

Figure 6.3: An example of a user review

The (anonymous) data collected from Gogobot surpassed in total over 300,000 ratings by 1759 users. After summarizing attraction ratings to destinations and after filtering out ratings on destinations not part of our database, 53028 remained. The tribes popularity for the 685 destinations were also recorded, and used in our Knowledge Based algorithm (see 4.3).

### 6.3 Travel costs

For making sure recommendations match a user’s budget preferences, some data on travel costs were needed. Daily expenses on location are highly dependent on the choice of hotel, food and activities. They are hard to predict, although importing information on possible hotels and restaurants would certainly help. Another approach would be to use statistics available on the cost-of-living per country or even per city. These price index can be found from different resources, for instance from [32].

In TravelWithFriends, the focus was set to the transportation costs, allowing the user to choose how much money he was willing to spend on getting to his destination. Various API’s and web services allow to lookup real-time prices for trains (like iRail[3], giving information on the Belgian trains), airplanes (for instance Google’s QPX Express API [39]) and various other
6.3 Travel costs

means of transport. A lot of different factors have to be taken into account if our system wanted to use these data sources, including but not limited to: finding the most suitable airport that flies to the wanted destination, combining different means of transport and predicting average prices in the future. Luckily, the webservice Rome2rio has done all these things and presented an API that can give us exactly the information we need in TravelWithFriends.

![Rome2rio web interface](image)

The Rome2rio API allows to look up transportation means and their predicted cost from any location in the world to any destination. It taps into the information of many different databases and APIs to gather information on flights, trains, busses, boats and even taxi prices to come up with all possible means to make the trip to your destination. Figure 6.4 shows part of the website interface. Figure 6.5 shows part of the JSON response from the API.

The TravelWithFriends database only stores the top itinerary presented by the Rome2rio API, as this service has already a good algorithm that puts the most relevant option first (based on duration and cost). The service provides with much more rich information that could also be used to present alternative routes or allow users to choose their preferred transportation mode. In the scope of this research, no further use is made of this information.
6.3 Travel costs

Figure 6.5: JSON response from API
Chapter 7

System design and implementation

In this chapter we give more details on the implementation and design of the TravelWithFriends web application. We start in Section 7.1 with a general view on the architecture of the application and the different components it encompasses. Next we’ll discuss each part in more detail, starting with the recommender engine in Section 7.2 and the lenskit-library it makes heavy use of. We show the implementation of the Grails web-application in Section 7.3 and its user interface in 7.4. Finally we show how the connection to the data storage was made in Section 7.5.

7.1 Client-server architecture

The high-level architecture of the server-side of our system is shown in Figure 7.1. We can see from the overview that our server-side has a three-layered architecture. The data layer takes care of storing and retrieving objects from the database. Most interactions of the web application with the database are handled by the domain classes, an inherited part of the Grails architecture. The recommender engine uses different DAOs to access the data directly. The service layer takes care of the business logic. In our application these are bundled in the recommender engine and accessed through service-classes. The application layer is the access point to and from the client side. The controllers guide the application, while the views take care of what is presented in the browser.
7.2 The recommender engine

The recommender engine is the place where all different recommendation approaches are gathered. It makes extensive use of the Lenskit\cite{12, 24} library, a open-source toolkit designed specially for building and researching recommender systems. Lenskit is written in Java and allows for a very modular approach to building recommender systems and choose which parts you want to customize, by making use of dependency injection. More info on how this works in depth can be found in \cite{12, 24}.

To build a recommender, first the configuration is stored in a LenskitConfiguration. Which parts were customized for each recommender can be seen from the configureRecommender method that produces this configuration. Each configuration for the different approaches will be discussed further in this section, but first we introduce some of the general ideas.

The core of a Lenskit recommender is in its ItemScorer class. This class scores items, i.e. based on the information available, it will predict how well a user will like this item. The higher the score, the better this item fits the user. Most ItemScorers need specific information to calculate these scores. A ModelBuilder will provide the Scorer with this information, packed as a Model object. The ItemScorer is then used by the Recommender class to select the final recommendation.

In our application the default recommender was used, the TopNItemRecommender. This item simply picks the $N$ highest scoring items and presents these as the final recommendation.

Another important part of each Recommender is the different DAOs (data access objects)
that provide the different classes with the data for the items (ItemDAOs), users (UserDAOs) and the ratings or other information saved (EventDAOs). An example of this structure is depicted in Figure 7.2 showing the class dependencies for the item-item recommender. With the general structure explained, we will now dive into each specific recommender class.

![Figure 7.2: Class hierarchy of the Item-Item recommender](image)

### 7.2.1 Item-Item recommender

First we show the configuration build of the item-item recommender approach. The class hierarchy was seen already in Figure 7.2. The `configureRecommender` class is depicted in Figure 7.3.

From Figure 7.3 we can distinguish the different building blocks of the item-item recommender. In Lenskit bindings are a way to specify which specific implementation of a building block you want to be used for this recommender. First the three DAOs are specified. Our data is stored in a MySQL database which we will access in Java through JDBC. For this reason a JDBCRatingDAO, a JDBCItemTitleDAO and a JDBCUserDAO were implemented to access the ratings, destinations’ information and users respectively.
7.2 The recommender engine

The recommender engine involves building an Item-item configuration. The ItemScorer was set to the ItemItemScorer with a neighborhood size of 20. This ItemItemScorer implements the algorithms discussed in Section 4.1. To score items, it makes use of the model provided by the ItemItemModelBuilder. In this case, the model holds the neighborhoods for each item: a mapping of each possible item-item pair to their similarity. These neighborhoods are calculated for the ItemItemScorer by the ConditionalProbabilitySimilarity class. This class contains the algorithm discussed in 4.1.2, using conditional probability of two items being rated by the same user as similarity measure.

7.2.2 Content-based recommender

The modularity of Lenskit allows us to easily swap out or reuse parts we have build for other recommenders. This can be seen from the builder class of the content-based recommender implementation (see Figure 7.4). Both the same JDBCItemTagDAO and JDBCUserDAO are used as access objects to the ratings and users. An adjusted version of the ItemDAO is used here, the JDBCItemTagDAO, which stores not only the different destinations but also their
**7.2 The recommender engine**

**Tag frequencies** (more information on the different data tables will be discussed in Section 7.5). The big difference lays in the ItemScorer used. The **TFIDFItemScorer** uses TF-IDF terms to characterize each destination. The final score for an item is the similarity between the item’s tag profile and the tag profile of the user. Here, cosine similarity is used as similarity measure. More details on these algorithms were discussed in 4.2.2.

```java
public class CBFBuilder {
    private LenskitConfiguration config;
    private Recommender cbfRecommender;

    public CBFBuilder(LenskitConfiguration config) throws RecommenderBuildException {
        this.config = config;
        this.cbfRecommender = LenskitRecommender.build(config);
    }

    public static LenskitConfiguration configureRecommender() {
        LenskitConfiguration config = new LenskitConfiguration();
        // configure the rating data source
        config.bind(EventDAO.class).to(JDBCRatingDAO.class);
        // specify item DAO implementation with tags
        config.bind(ItemDAO.class).to(JDBCItemTagDAO.class);
        // our user DAO can look up by user name
        config.bind(UserDAO.class).to(JDBCUserDAO.class);

        // use the TF-IDF scorer to score items
        config.bind(ItemScorer.class).to(TFIDFItemScorer.class);
        return config;
    }
}
```

Figure 7.4: Content-based configuration builder

### 7.2.3 Knowledge-based recommender

The knowledge-based approach again reuses the same implementations for **JDBCUserEventDAO** and **JDBCUserDAO** as in the item-item recommender. In the two previous configurations no **UserEventDAO** was specified because the default implementation was sufficient (the default implementation just takes all ratings for one user from the global **RatingDAO**). Here we implemented an augmented **UserEventDAO**, the **JDBCUserEventKBDAO** that gives one extra method: `getUserProfile(long id)` that returns the interests profile of a user. The **KBItemScorer** combines these interests and the user’s ratings to a final tourist profile (see 4.3.2). Similarity between users and destinations are here calculated again with the cosine similarity.
7.2 The recommender engine

7.2.4 The RecommenderManager class

An important piece of the puzzle in the TravelWithFriends application is the RecommenderManager class. It functions as a connection point between the Recommender Engine and the Services of the web application (see Figure 7.1). The class is a facade, allowing to call one simple function to get a recommendation, while hiding the implementation details of these recommenders. As such it has an access method to each of the recommender approaches discussed above (recommendCF, recommendCBF and recommendKB), and additionally four extra methods:

- **setConstraints**: executes a hard filter that reduces the possible destinations to be recommended, based on the constraints chosen by the user

- **recommendHYB**: the hybrid recommender, constructed by combining the results of the three base recommendation algorithms

- **recommendTop**: a 'Most Popular' recommendation, returning the most popular destinations not yet rated by the user
• *recommendGroup*: returning a group recommendation for a given list of users

### 7.3 The Grails web application

The TravelWithFriends web application was built with Grails[45], an open-source web framework written in the Groovy programming language and built on top of well-supported technologies like Java EE, Spring MVC, Hibernate, SiteMesh and Gradle. Figure 7.6 shows us the structure of the web application. The structure shows clearly that our web application makes use of the Model-View-Controller (MVC) software architecture pattern. The Views take care of the presentation to the user. In Grails, these are stored in GSPs (Groovy Server Pages). The Controller classes take care of the application flow, process the user inputs and request the needed information from the Services and Domains and provide these to the Views. The model component of MVC is spread over two components in the Grails context: the Domain classes are direct representations of data stored in the database. The Service classes contain all the business logic to keep the application running. We’ve seen in the previous section that it will be connected to the recommender engine via the RecommendationManager and will provide those to the controllers.

![Figure 7.6: The web application structure](image)

### 7.3.1 The Domain Classes

In a Grails web application, all Domain classes and their relations are mapped to a database through GORM: Grails’ object relational mapping (ORM) implementation[33] which runs Hibernate 3 underneath. Our web application contains following Domain classes:
7.3 The Grails web application

- **User**: representing a user’s information including his username, (encrypted) password and his **UserProfile**

- **UserProfile**: connected to each **User** is a **UserProfile**, containing his last indicated interests for the different categories like history, adventure, nature etc.

- **Destination**: representing a travel destination, including his name, geographic location, country and continent

- **Rating**: contains a score by a **User** for a certain **Destination**

- **TravelGroup**: represents a group of **Users** that wish to travel together. Has a name, creator and invitation code

- **TravelGroupRec**: linked to a **TravelGroup**. Contains the list of 10 **Destinations** recommended to the **TravelGroup** as well as the specific constraints that were set for the recommendation. This domain also contains a list of **DestinationRanks** set by users for this top 10 list

- **DestinationRank**: represents a ranking of a **User** for a **Destination**. Linked to a **TravelGroupRec**

More precise information on what fields each **Domain class** has will will be presented in 7.5.

7.3.2 The Controllers

As mentioned before, the **Controller** classes take care of the application flow, processes the user inputs, requests the needed information from the **Services** and **Domains** and provide these to the **Views**. Our web application has 7 controllers, each with their own specific focus on a part of our application. Four of these controllers are connected to a specific domain: the **UserController**, **RatingController**, **DestinationController** and **TravelGroupController**. These controllers take care of the CRUD operations for that domain. They mostly manage four views: an **index** (=READ ALL), **show** (=READ), **create** (=CREATE) and **edit** (=UPDATE,DELETE).

Besides these we also have three controllers dedicated to functionalities of the web application: the **LoginController** takes care of logging in and out of users and the registration flow, the
RecommenderController leads the flow of an individual recommendation and the GroupRecommenderController takes care of the flow for group recommendations.

### 7.3.3 The Services

The services take care of all business logic and present the controllers with a simple interface to access these. Currently, the only service in our application is a RecommenderService. Since we already have a facade to all recommender logic (the RecommendationManager, see 7.2.4), this class only need to form a bridge between the controller and the RecommendationManager.

### 7.3.4 The Views

The views are the collection of web pages the user can access. These pages will be discussed in the following section when we talk about the user interface.

### 7.4 User interface

In this section we will show some of the interesting user interactions available in the TravelWithFriends application. The web application is fully functional and as such features many pages. Some are less interesting to discuss like the register page, creating and joining groups and we will not show them here. We will however focus on three different functions of the application: 1) finding destinations and adding ratings, 2) receive a personalized, individual recommendation and 3) arrange a group recommendation.

#### 7.4.1 Finding destinations and adding ratings

For a user to add ratings to his profile, he can choose from 2 options. The first is to look up the destination from a drop-down list, choose his score and add the rating (see Figure 7.7). All his ratings are collected and shown on the right side of the screen.
As a visual aid to find new destinations to rate, a second option was made with Google Maps API[20] as can be seen in Figure 7.8. The map allows for easy navigation and when an interesting location is found, the user can press the ‘Add Rating’ link to open a new tab with the page to add the rating, with the correct destination already selected. This could be made even more user-friendly by combining both pages to a single-page solution (for instance with AJAX).
7.4.2 Individual recommendation

Next, we show how a user can receive a personal recommendation in TravelWithFriends. After clicking the link ‘Individual recommendation’ on the home page, the user is brought to the ‘Interests Selection’-page where he can indicate his affinity with the 19 available tourist profiles (see Figure 7.9). A user can choose to tick off the groups he has an opinion towards, moving the slider both above (positive interest) or below (negative interest) the 50% mark. Unaltered values are saved as 50%. These values are stored in a user’s profile and are used in the Knowledge Based recommender approach to find matching destinations and predict ratings.

Next, a page to select some constraints and filters is presented to the user (see Figure 7.10). This page allows to choose the transport budget, the minimum and maximum distance to travel and their home location, their preferred continents and some activities they want to be available. All of the options can be ticked on or off, allowing the user to choose what filters he wants to use. The options selected are then used in the Content-Based recommender approach as well as to set some hard constraints to reduce the possible destinations recommended.

Finally the user would be presented with his personalized recommendations. During this research, 5 different recommendation approaches were compared to each other. The user was thus presented with five lists of 8 destinations each, who he could then evaluate (see Chapter 8).
7.4 User interface

7.4.3 Group recommendation

To start a group recommendation, first a user has to join an existing group or create a new one through the appropriate pages (found in the top bar). He can then click to begin group recommendation, select the constraints of the request and start the recommendation. After merging the profiles and ratings of the group members (see also Chapter 5) he is then presented with the ten candidates destinations for the group. In this screen, the user give a ranking to the destinations he likes most, leaving the other options open, as shown in Figure 7.11.

![Figure 7.10: Choose filters and constraints page](image1)

![Figure 7.11: Initial ten candidates for the group recommendation](image2)
After finishing his selection, the user is brought to the group recommendation page where the **final ranking** is shown (see Figure 7.12). The list shown is only the result of the rankings of the users that have given in their ranked list. Not all users will do that at the same time, so it is quite possible the list shown is not the final result. On the right side of that same page, a **list of group members** is shown together with their **status**: a red cross indicates they haven’t submitted their ranked list, a green checkmark shows if they have.

![Figure 7.12: Final selection and ranking for the group](image)

**7.5 Data model and storage**

The data in the application was all stored in a MySQL database for easy and quick access. The tables in the database are closely related to the domains discussed previously or hold some extra information on the destinations to help the recommendations. The full **Entity-Relationship Model (ERM)** can be seen in Figure 7.13.

The **Destination** and **User** tables are at the heart of the model. **Ratings** are linked to a unique (user,destination)-pair and hold the score (value between 1 to 5). Besides the ratings, a destination is also connected to three extra information sources. The first is the **TransportCost** table that holds the price and duration to travel from a destination to another. Secondly we have stored the **Tags** of each destination together with its count. Last, there’s the **DestinationProfile** holding 19 values, each value represents the affinity of the destination with a profile category like Adventure,
Another large part of the database is dedicated to storing information for group recommendations. The *TravelGroup* entity stores information for a group, and has one or more group members. To start a group recommendation, a *TravelGroupRec* element is created, storing the shortlist of destinations (these are the 10 recommended locations from the group profile) and the constraints that were set. Finally each user can give a ranking for each of those 10 locations, which are stored as the quadruplet (user_id, destination_id, group_rec_id, ranking).
Chapter 8

Evaluation

This chapter presents how the TravelWithFriends application was evaluated. We start by explaining the evaluation method and test setup in Section 8.1. The evaluation tests three different aspects of our application, and the following sections each focus on a different aspect. Section 8.2 talks about the general experiences of our test users. In Section 8.3 the different recommender approaches are compared and evaluated. Section 8.4 does the same but for the group recommender. Finally Section 8.5 summarized the results and mentions the takeaways from this chapter.

8.1 Test setup and questionnaire

The TravelWithFriends application was presented to a group of 16 users for evaluating the design of the recommender and the different recommendation algorithms. The users were observed while using the application on a local system. An approach that allowed for a much richer form of feedback then when test users were invited to use a web application from distance.

For collecting quantitative feedback, a questionnaire was answered by each test user. These questions were inspired by the research of Pu [38] in which she set forward 15 different domains for evaluating a recommender system and 30 possible questions to ask. The evaluation was divided into three domains, a general evaluation, evaluation of the different recommender algorithms and a part on the group recommendation. Following sections will present the results in each domain.
8.2 General evaluation of the individual recommender

After testing the individual recommender system, the test users were invited to answer four multiple choice questions to measure their satisfaction with the system. The possible answers were: disagree, slightly disagree, neutral, agree and totally agree.

1. Overall, I am satisfied with the recommender.
2. I found it easy to tell the system what I like/dislike
3. I am convinced of the items recommended to me
4. In case TravelWithFriends becomes online available, I would consider using this recommender for finding travel destinations

Further, they were asked to answer two open-ended questions regarding the features available.

- What feature or selection option did you find most important to decide on your recommendation?
- Which features or selection options did you feel were missing?

The results from the test group showed (see Figure 8.1) that all 16 users were overall satisfied with the recommender (Question 1), with 14 users agreeing and 2 ”Totally” agreed. They were pleased with the application’s ability to show them new destinations they didn’t know of before and found the experience an enjoyable way of finding their next travel destinations. Almost all users also indicated they would use the application if it became available online (Question 4).

The results also showed the users were more divided to whether it was easy enough to tell their preferences by the system (Questions 2). This shows the application has yet to improve their user interface and broaden the inputs by the user. Some suggestions were given in the 2nd open question, were some test users answered they would like more choices in choosing their type of vacation (think: citytrip vs. tour vs. hiking trip), the option of a general safety advice and tourist-friendliness of the destination and more activities to choose from.
Question 3 was closely related to Question 2: users were also not all sure that the system had correctly interpreted the preferences they gave to the system. This could be due to the lack of some options, but is definitely also a sign the application needs more explanations to the user why this items were presented to them. A smarter user interface and more informed final result should help to solve this.

8.3 Evaluation of the different recommender approaches

The second part of the evaluation dealt with the four different algorithms presented in Chapter 4: the collaborative filtering (CF), content-based filtering (CBF), knowledge-based recommender (KB) and the hybrid recommender (HYB). As a baseline to compare the different algorithms with, a fifth approach was implemented which simply returned the top most-popular destinations (TOP). The TOP-algorithm returned the most rated destinations on TripAdvisor. In this case, destinations already rated by the user were also removed from the candidates to recommend.


8.3 Evaluation of the different recommender approaches

<table>
<thead>
<tr>
<th>Approach</th>
<th>1st</th>
<th>2nd</th>
<th>3th</th>
<th>4th</th>
<th>5th</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Most popular (TOP)</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>3.81</td>
</tr>
<tr>
<td>Content-based (CBF)</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2.81</td>
</tr>
<tr>
<td>Knowledge-based (KB)</td>
<td>4</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>2</td>
<td>2.81</td>
</tr>
<tr>
<td>Collaborative (CF)</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>3.25</td>
</tr>
<tr>
<td>Hybrid (HYB)</td>
<td>6</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>3</td>
<td>2.31</td>
</tr>
</tbody>
</table>

Table 8.1: Given rankings for the different approaches

Users were invited to use the web application as they pleased and were asked to then start a individual recommendation. After finishing the different preparatory steps (adding ratings, selecting interests and choosing constraints), the user was presented with five different lists of 8 recommendations each. These five lists were shuffled randomly and presented without any hint of the exact algorithm used, to assure unbiased results when evaluating the lists.

The test user was first asked to rank the five lists in order of his liking of the presented recommendations of each list. Table 8.1 shows the results from this question. Figure 8.2 visualizes the distribution of these rankings. From the image it is clear that the hybrid recommender was the favorite overall, with 6 users choosing this approach as their top list, and 5 more put it in 2nd place. Content-based and Knowledge-based were also generally liked, while the TOP-approach scored the worst.

Given the small test sample of only 16 users, a small statistical analysis was done for the average ranks of the different approaches in the form of a Student’s t-test. The null-hypothesis was that the difference in mean were merely due to randomness of the results. The results (see Appendix A) showed that the difference in mean were statistically significant between the baseline (TOP-algorithm) and the hybrid recommender (p-value: 0.004), the content-based (p-value: 0.028) and the knowledge-based approach (p-value: 0.031). Only the collaborative recommender did not show statistical evidence of being higher evaluated (p-value: 0.251).

Next, each user was then asked to answer four questions for each recommended list. As possible answer to each question the tester had to pick again from the 5 same possible answers: disagree, slightly disagree, neutral, agree and totally agree. The questions were as follows:
8.3 Evaluation of the different recommender approaches

Figure 8.2: Distribution of final rankings given to the recommenders

Q1. **[Match interests]** The destinations recommended to me matched my interests

Q2. **[Novelty]** The recommender system helped me discover new destinations to visit

Q3. **[Diversity]** The destinations recommended to me are diverse

Q4. **[Usefulness]** The recommender gave me useful suggestions

Full results can be found in Appendix A. We do present here the average answer for each question in Figure 8.3. Question 1 shows that the hybrid and the knowledge-based approaches match the user’s interests best. The knowledge-based recommender’s primary contributing factor is the interests profile made by the user. This clearly has an impact on the capability of the recommender to match destinations to the user’s interests.

Unsurprisingly, the TOP-recommender scores very low on the novelty question. The Collaborative recommender outscores the other approaches here. This phenomenon was already noticed in previous research: collaborative filters allow for unexpected recommendations because users can be very versatile in their rating behavior. The content- and knowledge-based approach miss this feature as they will only recommend destinations similar to the user’s previous experiences. The low score of the hybrid approach is probably due to mixing the different approaches balancing
8.4 Evaluation of the group recommender

Out the unexpected recommendations and ending up with only the more obvious choices. This result requires more research into its underlying reasons however.

Question 3, regarding the diversity of the recommendations again unsurprisingly scores worst for the TOP-approach. Content-based recommenders are often blamed a lack of diversity in their recommendations and the test users of TravelWithFriends have also noticed that in our system, while the knowledge-based recommender scores best on this domain.

Question 4 regarding the usefulness of the recommender can be seen as a general review of how good the recommendations were. The results show they are in line with the rankings given earlier: hybrid, knowledge-based and content-based score best, with the collaborative approach lacking behind and the baseline TOP-approach regarded as the least useful.

8.4 Evaluation of the group recommender

Our test users were also presented to the group recommender. The 16 test users formed four groups, two groups could be described as a family group, with parents and siblings put together. One group consisted of four family members, the other of three. The two other groups were
groups of friends, sized six and three people respectively. The users were presented with the ten candidates, ranked these as they pleased and then received a final recommendation of five top destinations (see Chapter 5). They were then asked to answer the four following questions:

**G1.** I understood why the destinations were recommended to our group

**G2.** The group recommendation was balanced and fair for all group members

**G3.** My opinion was sufficiently heard to make the recommendation to the group

**G4.** I am satisfied with the group recommendation.

The results in Figure 8.4 show the users are very satisfied with the group recommender and feel their own opinion was sufficiently heard to make the final recommendation. They are however uncertain to what these results were based on and whether they are a fair result for the whole group. Just like the individual recommender, more explanations and feedback from the application to the users is wished for.

### 8.5 Conclusion

From the results we can conclude that the users were fairly happy with both the individual and the group recommender. They enjoyed the new way of discovering destinations and were always happy to find at least a few interesting new places to consider. The results have also shown that a hybrid recommender gives the most pleasing result, while the three separate recommender approaches each have their merits and limitations. The user has also clearly shown they want more feedback from the system and clearer explanations to accompany the recommendations made.
Figure 8.4: Results of the questions regarding the group recommender.
Chapter 9

Conclusion

9.1 Conclusion

This work has presented the research and design process to making a travel recommender system for groups. First, the existing recommender systems and previous research on the topic were studied. The field of research on recommender systems proved already very broad, with many different algorithms and strategies already presented. Each of these algorithms had their merits and limitations. The domain of travel recommendations was much less explored with only a few attempts at making a recommender system for this domain. This void presented some opportunities for this work to fill in some of the gaps.

Because travel destinations proves to be a complex domain for recommendations, no single algorithm would be able to give the necessary depth to make these recommendations. A hybrid system that takes some of the best ideas of different approaches was proposed next. The proposed design consisted of a collaborative, a content-based and a knowledge-based component. The collaborative filtering approach makes recommendations to the user purely based on his rating profile. This recommender looks for agreements in rating profile between the user and other users to find destinations to present. Finding a good data set of ratings proved quite the challenge, but some creativity and a community-driven travel site called Gogobot proved the right source.

The content-based recommender takes user-applied tags of the different attractions of a destina-
tion and uses theses to summarize each location. This approach looks to recommend destinations with similar tags as the user’s highest rated destinations. Finally a knowledge-base was established for making the third recommender. Different important elements in deciding for a next travel destination were brought together including the transport costs, available attractions and the distance and location of the destination. More information sources had to be found in order to collect these. The WikiVoyage and GeoLocations open-source projects proved to be valuable sources of information for finding descriptions of destinations and their geo-information. For calculating transport costs, the API of Rome2Rio was accessed. Then all that was left was bringing these three recommenders together into one final hybrid system. In this system an additional pre-filtering step was brought in to remove any impossible destinations from the candidates to recommend to the user.

With the individual recommender design ready, an extension was presented next to also help out groups in finding their next vacation destination. This group recommender makes use of the hybrid recommender to select a first preliminary list of ten candidates based on the profiles of all group members. A second step allows the group members to choose their favorites from the list by giving them all a ranking. A BORDA count would finally decide on what top 5 locations would fit the group best.

The TravelWithFriends application was then presented, a web application that implements both the individual and the group recommender design. This application was build with the Grails web framework and makes extensive use of the Lenskit library for its different recommender implementations. The prototype was then presented to a group of 16 test users to evaluate the different assumptions made previously in this work. Their reactions showed that the application has plenty of interesting features. The different recommender approaches were compared next, where the hybrid recommender came out on top of the results, with all three separate approaches also showing their worth.

9.2 Future Work

As is the case in many research, making this work has unveiled a lot of new questions and further research opportunities. The three different recommender approaches have each proven
to bring valuable information into the recommender. Further research is needed into how these recommenders optimally work together and what weights each recommender should be given. The knowledge-based approach also leaves space for many more extensions, of which some were brought forth by the test users: a wider choice in activities, a safety measure and choosing durations of the travel plans are welcomed as extra features.

Another interesting path to research is the extension of the destination model. The TravelWith-Friends model is limited to popular tourist attraction sites and cities. An improved model should also be able to recommend a group of close-by locations, a tour or a region to explore. Further a strategy must be found to present the user with more feedback on his recommendation request, allowing him to tweak some settings and request a new recommendation. And the test users have also indicated a lack of transparency and explanations of the system, which is another point where the user interface can be improved.

Finally the evaluation of the group recommender also showed that more feedback and explanations towards the group recommendation is needed. An extended feedback process beyond the ranking of the initial list of ten candidates could also prove an interesting path for further research.
Appendix A

Evaluation results and questionnaire

A.1 Evaluation results

<table>
<thead>
<tr>
<th></th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Totally Agree</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>The destinations recommended to me matched my interests</td>
<td>6.25% 1</td>
<td>25.00% 4</td>
<td>18.75% 3</td>
<td>37.50% 6</td>
<td>12.50% 2</td>
<td>16</td>
</tr>
<tr>
<td>The recommender system helped me discover new destinations to visit</td>
<td>18.75% 3</td>
<td>31.25% 5</td>
<td>18.75% 3</td>
<td>25.00% 4</td>
<td>6.25% 1</td>
<td>16</td>
</tr>
<tr>
<td>The destinations recommended to me are diverse</td>
<td>12.50% 2</td>
<td>31.25% 5</td>
<td>31.25% 5</td>
<td>25.00% 4</td>
<td>0.00% 0</td>
<td>16</td>
</tr>
<tr>
<td>The recommender gave me useful suggestions</td>
<td>0.00% 0</td>
<td>12.50% 2</td>
<td>56.25% 9</td>
<td>31.25% 5</td>
<td>0.00% 0</td>
<td>16</td>
</tr>
</tbody>
</table>

Figure A.1: Full results questionnaire for TOP
### A.1 Evaluation results

<table>
<thead>
<tr>
<th>Statement</th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Totally Agree</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>The destinations recommended to me matched my interests</td>
<td>0.00%</td>
<td>12.50%</td>
<td>31.25%</td>
<td>50.00%</td>
<td>6.25%</td>
<td>16</td>
</tr>
<tr>
<td>The recommender system helped me discover new destinations to visit</td>
<td>0.00%</td>
<td>6.25%</td>
<td>12.50%</td>
<td>56.25%</td>
<td>25.00%</td>
<td>16</td>
</tr>
<tr>
<td>The destinations recommended to me are diverse</td>
<td>6.25%</td>
<td>12.50%</td>
<td>31.25%</td>
<td>37.50%</td>
<td>12.50%</td>
<td>16</td>
</tr>
<tr>
<td>The recommender gave me useful suggestions</td>
<td>0.00%</td>
<td>6.25%</td>
<td>37.50%</td>
<td>43.75%</td>
<td>12.50%</td>
<td>16</td>
</tr>
</tbody>
</table>

Figure A.2: Full results questionnaire for CBF

<table>
<thead>
<tr>
<th>Statement</th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Totally Agree</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>The destinations recommended to me matched my interests</td>
<td>0.00%</td>
<td>6.25%</td>
<td>18.75%</td>
<td>43.75%</td>
<td>31.25%</td>
<td>16</td>
</tr>
<tr>
<td>The recommender system helped me discover new destinations to visit</td>
<td>0.00%</td>
<td>0.00%</td>
<td>31.25%</td>
<td>56.25%</td>
<td>12.50%</td>
<td>16</td>
</tr>
<tr>
<td>The destinations recommended to me are diverse</td>
<td>0.00%</td>
<td>0.00%</td>
<td>25.00%</td>
<td>56.25%</td>
<td>18.75%</td>
<td>16</td>
</tr>
<tr>
<td>The recommender gave me useful suggestions</td>
<td>0.00%</td>
<td>6.25%</td>
<td>25.00%</td>
<td>56.25%</td>
<td>12.50%</td>
<td>16</td>
</tr>
</tbody>
</table>

Figure A.3: Full results questionnaire for KB
### A.1 Evaluation results

#### Figure A.4: Full results questionnaire for CF

<table>
<thead>
<tr>
<th></th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Totally Agree</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>The destinations recommended to me</td>
<td>0.00%</td>
<td>0</td>
<td>56.25%</td>
<td>25.00%</td>
<td>12.50%</td>
<td>16</td>
</tr>
<tr>
<td>matched my interests</td>
<td></td>
<td></td>
<td>9</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.00%</td>
<td>0</td>
<td>6.25%</td>
<td>1</td>
<td>0.00%</td>
<td>16</td>
</tr>
<tr>
<td>The recommender system helped me to</td>
<td></td>
<td></td>
<td>9</td>
<td>4</td>
<td>12.50%</td>
<td></td>
</tr>
<tr>
<td>discover new destinations to visit</td>
<td>0.00%</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>12.50%</td>
<td>2</td>
<td>6.25%</td>
<td>1</td>
<td>0.00%</td>
<td>16</td>
</tr>
<tr>
<td>The destinations recommended to me</td>
<td></td>
<td></td>
<td>1</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>are diverse</td>
<td>0.00%</td>
<td>0</td>
<td>43.75%</td>
<td>3</td>
<td>18.75%</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The recommender gave me useful</td>
<td>0.00%</td>
<td>0</td>
<td>18.75%</td>
<td>3</td>
<td>18.75%</td>
<td>16</td>
</tr>
<tr>
<td>suggestions</td>
<td></td>
<td></td>
<td>3</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Figure A.5: Full results questionnaire for HYB

<table>
<thead>
<tr>
<th></th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Totally Agree</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>The destinations recommended to me</td>
<td>0.00%</td>
<td>0</td>
<td>6.25%</td>
<td>50.00%</td>
<td>31.25%</td>
<td>16</td>
</tr>
<tr>
<td>matched my interests</td>
<td></td>
<td></td>
<td>1</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.00%</td>
<td>0</td>
<td>18.75%</td>
<td>3</td>
<td>12.50%</td>
<td>16</td>
</tr>
<tr>
<td>The recommender system helped me to</td>
<td></td>
<td></td>
<td>3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>discover new destinations to visit</td>
<td>0.00%</td>
<td>0</td>
<td>25.00%</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.00%</td>
<td>0</td>
<td>6.25%</td>
<td>1</td>
<td>18.75%</td>
<td>16</td>
</tr>
<tr>
<td>The destinations recommended to me</td>
<td></td>
<td></td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>are diverse</td>
<td>0.00%</td>
<td>0</td>
<td>25.00%</td>
<td>9</td>
<td>12.50%</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The recommender gave me useful</td>
<td>0.00%</td>
<td>0</td>
<td>12.50%</td>
<td>3</td>
<td>18.75%</td>
<td>16</td>
</tr>
<tr>
<td>suggestions</td>
<td></td>
<td></td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A.2 Student’s T-test for mean rankings

T-Test Hybrid - Top

<table>
<thead>
<tr>
<th>Group Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>N</strong></td>
</tr>
<tr>
<td>rank</td>
</tr>
<tr>
<td>hybrid</td>
</tr>
<tr>
<td>top</td>
</tr>
</tbody>
</table>

---

**Independent Samples Test**

<table>
<thead>
<tr>
<th></th>
<th>Levene’s Test for Equality of Variance</th>
<th>t-test for Equality of Means</th>
<th>95% Confidence Interval of the Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>Sig</td>
<td>df</td>
</tr>
<tr>
<td>rank</td>
<td>0.457</td>
<td>0.976</td>
<td>28.349</td>
</tr>
<tr>
<td></td>
<td>0.466</td>
<td>0.994</td>
<td>30</td>
</tr>
</tbody>
</table>

Figure A.6: T-test mean of Hybrid and Top recommender

T-Test Content-Based & Top

<table>
<thead>
<tr>
<th>Group Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>N</strong></td>
</tr>
<tr>
<td>rank</td>
</tr>
<tr>
<td>ref</td>
</tr>
<tr>
<td>top</td>
</tr>
</tbody>
</table>

---

**Independent Samples Test**

<table>
<thead>
<tr>
<th></th>
<th>Levene’s Test for Equality of Variance</th>
<th>t-test for Equality of Means</th>
<th>95% Confidence Interval of the Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F</td>
<td>Sig</td>
<td>df</td>
</tr>
<tr>
<td>rank</td>
<td>0.036</td>
<td>0.851</td>
<td>29.764</td>
</tr>
<tr>
<td></td>
<td>0.036</td>
<td>0.851</td>
<td>29.764</td>
</tr>
</tbody>
</table>

Figure A.7: T-test mean of Content-based and Top recommender
A.2 Student’s T-test for mean rankings

T-Test Knowledge Based & Top

<table>
<thead>
<tr>
<th>Group</th>
<th>N</th>
<th>Mean</th>
<th>Std Deviation</th>
<th>Std Error Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>rank</td>
<td>16</td>
<td>2.91</td>
<td>1.32</td>
<td>.322</td>
</tr>
<tr>
<td>top</td>
<td>16</td>
<td>3.81</td>
<td>1.167</td>
<td>.292</td>
</tr>
</tbody>
</table>

Independent Samples Test

<table>
<thead>
<tr>
<th>Group</th>
<th>df</th>
<th>F</th>
<th>Sig</th>
<th>t</th>
<th>df</th>
<th>Sig (Stalled)</th>
<th>Mean Difference</th>
<th>Std Error Difference</th>
<th>95% Confidence Interval of the Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>rank</td>
<td>16</td>
<td>.009</td>
<td>.928</td>
<td>-2.263</td>
<td>36</td>
<td>.031</td>
<td>-1.000</td>
<td>.442</td>
<td>-1.900 - .997</td>
</tr>
</tbody>
</table>

Figure A.8: T-test mean of Knowledge-based and Top recommender

T-Test Collaborative - Top

<table>
<thead>
<tr>
<th>Group</th>
<th>N</th>
<th>Mean</th>
<th>Std Deviation</th>
<th>Std Error Mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>rank</td>
<td>16</td>
<td>3.25</td>
<td>1.028</td>
<td>.362</td>
</tr>
<tr>
<td>top</td>
<td>16</td>
<td>3.81</td>
<td>1.167</td>
<td>.292</td>
</tr>
</tbody>
</table>

Independent Samples Test

<table>
<thead>
<tr>
<th>Group</th>
<th>df</th>
<th>F</th>
<th>Sig</th>
<th>t</th>
<th>df</th>
<th>Sig (Stalled)</th>
<th>Mean Difference</th>
<th>Std Error Difference</th>
<th>95% Confidence Interval of the Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>rank</td>
<td>16</td>
<td>3.007</td>
<td>.04</td>
<td>-1.170</td>
<td>30</td>
<td>.251</td>
<td>-1.553</td>
<td>.491</td>
<td>-1.544 - .410</td>
</tr>
</tbody>
</table>

Figure A.9: T-test mean of Collaborative and Top recommender
A.3 Questionnaire

Global

1. Overall satisfaction system

<table>
<thead>
<tr>
<th></th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overall, I am satisfied with the recommender.</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>I found it easy to tell the system what I like/dislike</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>I am convinced of the items recommended to me</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>In case TravelWithFriends becomes online available, I would</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>consider using this recommender for finding travel destinations</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
</tbody>
</table>

2. What feature/selection option did you find most important to decide on your recommendation

3. Which feature/selection options did you feel were missing

Different algorithms

4. Rank the different Lists

<table>
<thead>
<tr>
<th>List 1</th>
<th>List 2</th>
<th>List 3</th>
<th>List 4</th>
<th>List 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5. Feedback List 1

<table>
<thead>
<tr>
<th></th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>The destinations recommended to me matched my interests</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The recommender system helped me discover new destinations to visit</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The destinations recommended to me are diverse</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The recommender gave me useful suggestions</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
</tbody>
</table>
6. Feedback List 2

<table>
<thead>
<tr>
<th></th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>The destinations recommended to me matched my interests</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The recommender system helped me discover new destinations to visit</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The destinations recommended to me are diverse</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The recommender gave me useful suggestions</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
</tbody>
</table>

7. Feedback List 3

<table>
<thead>
<tr>
<th></th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>The destinations recommended to me matched my interests</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The recommender system helped me discover new destinations to visit</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The destinations recommended to me are diverse</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The recommender gave me useful suggestions</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
</tbody>
</table>

8. Feedback List 4

<table>
<thead>
<tr>
<th></th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>The destinations recommended to me matched my interests</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The recommender system helped me discover new destinations to visit</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The destinations recommended to me are diverse</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The recommender gave me useful suggestions</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
</tbody>
</table>

9. Feedback List 5

<table>
<thead>
<tr>
<th></th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>The destinations recommended to me matched my interests</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The recommender system helped me discover new destinations to visit</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The destinations recommended to me are diverse</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
<tr>
<td>The recommender gave me useful suggestions</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
<td>c</td>
</tr>
</tbody>
</table>
## 10. Group recommendation

<table>
<thead>
<tr>
<th></th>
<th>Disagree</th>
<th>Slightly Disagree</th>
<th>Neutral</th>
<th>Agree</th>
<th>Totally Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>I understood why the destinations were recommended to our group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>The group recommendation was balanced and fair for all group members</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>My opinion was sufficiently heard to make the recommendation to the group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>I am satisfied with the group recommendation.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bibliography


List of Figures

3.1 Overview TravelWithFriends application ........................................... 18
3.2 Data flow and the recommender engine ............................................. 19

4.1 TripAdvisor(R) page and attraction information .................................. 29
4.2 Choose filters and contraints page ................................................... 32

5.1 Different steps in the group recommender process ............................... 35

6.2 Gogobot’s tribes page (http://www.gogobot.com/tribes) ......................... 41
6.3 An example of a user review ...................................................... 42
6.4 Rome2rio web interface .......................................................... 43
6.5 JSON response from API ......................................................... 44

7.1 Three-layered architecture of our system ........................................... 46
7.2 Class hierarchy of the Item-Item recommender .................................... 47
7.3 Item-item configuration builder ...................................................... 48
7.4 Content-based configuration builder ............................................... 49
7.5 Knowledge-based configuration builder ............................................ 50
7.6 The web application structure .......................................................... 51
7.7 Adding a new rating ................................................................. 54
7.8 Find a destination to add rating ..................................................... 54
7.9 Indicate interests page ............................................................... 55
7.10 Choose filters and contraints page ................................................ 56
LIST OF FIGURES

7.11 Initial ten candidates for the group recommendation ........................................ 56
7.12 Final selection and ranking for the group .......................................................... 57
7.13 Entity-relationship model ..................................................................................... 58

8.1 Results of the general questions. ............................................................................. 61
8.2 Distribution of final rankings given to the recommenders ........................................ 63
8.3 Average answer on questions different algorithms .................................................. 64
8.4 Results of the questions regarding the group recommender ..................................... 66

A.1 Full results questionnaire for TOP ........................................................................ 70
A.2 Full results questionnaire for CBF .......................................................................... 71
A.3 Full results questionnaire for KB ............................................................................ 71
A.4 Full results questionnaire for CF ............................................................................ 72
A.5 Full results questionnaire for HYB .......................................................................... 72
A.6 T-test mean of Hybrid and Top recommender ........................................................ 73
A.7 T-test mean of Content-based and Top recommender .............................................. 73
A.8 T-test mean of Knowledge-based and Top recommender ........................................ 74
A.9 T-test mean of Collaborative and Top recommender ............................................... 74
List of Tables

4.1 User rating profile .................................................. 24
4.2 User binary rating profile ............................................ 25
5.1 Group scoring example ................................................ 36
5.2 Group ranking example with BORDA count ....................... 37
8.1 Given rankings for the different approaches ....................... 62