



UNIVERSITÀ DEGLI STUDI  
DI TRENTO

---

DEPARTMENT OF INFORMATION ENGINEERING AND COMPUTER SCIENCE  
**ICT International Doctoral School**

# EFFECTIVE RECOMMENDATIONS FOR LEISURE ACTIVITIES

Beatrice Valeri

Advisors:

Prof. Fabio Casati and Dr. Florian Daniel

Università degli Studi di Trento

---

December 2015



# Aknowledgments

First of all, I thank my advisors and colleagues in Trento, which guided and supported me in the last years. We helped each other and found together a way to grow and learn the most from this experience together. I want to thank also the people I met in Grenoble during my internship, as they showed me how collaboration can be successful even if we just met and we had different backgrounds. They did not let me feel alone in my first long experience out of Italy and they motivated me to look for other experiences around the world.

Special thanks go to my husband. He always supported me during the hardest periods, trying to keep me up and helping every time he could. He was always there when I needed someone to talk to, to clarify my ideas and always kept me updated with the novelties in the world of mobile devices. He helped even when we needed more people supporting the data collection in some of the experiments we run.

My friends supported me keeping the leisure time funny and distracting me from work when I needed it.

My family and my relatives were always there for me and they were understanding when I had to postpone our meetings because I had some extra work to do.

*Beatrice*



# Abstract

*People nowadays find it difficult to identify the best places to spend their leisure time performing different activities. Some services have been created to give a complete list of the opportunities offered by the city in which they live, but they overload people with information and make it difficult for them to identify what is more interesting. Personalized recommendations partially solve this problem of overload, but they need a deeper understanding of the personal tastes of people and of the different ways in which people want to spend their leisure time.*

*In this thesis we identify the requirements for a recommender system for leisure activities, study which data are needed and which algorithm better identifies the most interesting options for each requester. We explore the effects of data quality on recommendations, identifying which kind of information is needed to better understand user needs and who can provide better-quality opinions.*

*We analyse the possibility of using crowdsourcing as a means for collecting ratings when volunteering is not providing the needed amount of ratings or when a new dataset of ratings is needed to answer some interesting research questions.*

*Finally, we show how the lessons learned can be applied in practice, presenting a prototype of personalized restaurant recommender service.*

## Keywords

Recommender systems, leisure activities, data quality, crowdsourcing.



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Requirements . . . . .	4
1.1.1	Learning user tastes . . . . .	5
1.1.2	Identifying the best restaurant . . . . .	7
1.1.3	Availability everywhere and anytime . . . . .	8
1.1.4	Starting the service . . . . .	9
1.2	Methodology . . . . .	10
1.3	Contributions and Results . . . . .	12
1.4	Structure of the Thesis . . . . .	14
1.5	List of Publications . . . . .	16
<b>2</b>	<b>State of the Art</b>	<b>19</b>
2.1	Discovering Places . . . . .	19
2.2	Recommending Items . . . . .	23
2.2.1	Main collaborative-filtering strategies . . . . .	24
2.2.2	Algorithms evaluation . . . . .	26
2.2.3	Recommending places . . . . .	28
2.2.4	Social networks and collaborative filtering . . . . .	29
2.3	Expressing Ratings . . . . .	31
2.3.1	Rating representation . . . . .	32
2.3.2	Collecting ratings . . . . .	34
2.4	Conclusion . . . . .	41

<b>3</b>	<b>The Role of Friends in Decision Making</b>	<b>43</b>
3.1	Introduction . . . . .	43
3.2	Sample Population and Comparisons with Baselines . . . . .	45
3.3	Formal Experiment Definition . . . . .	47
3.4	Definition of Recommendation Strategies . . . . .	48
3.5	Evaluation of the Different Recommendation Algorithms . . . . .	49
3.6	Conclusion . . . . .	50
<b>4</b>	<b>How Purpose Influences Opinion</b>	<b>53</b>
4.1	Introduction . . . . .	53
4.2	Formal Definitions . . . . .	54
4.3	Extending the Experiment of Chapter 3 . . . . .	56
4.4	Understanding Preferences Across Activities . . . . .	57
4.5	Effect of Purpose on User Preferences . . . . .	59
4.6	Conclusion . . . . .	64
<b>5</b>	<b>Purpose-orientation and Focus on Locals</b>	<b>65</b>
5.1	Introduction . . . . .	65
5.2	Background . . . . .	67
5.3	Method . . . . .	68
5.3.1	Data collection . . . . .	68
5.3.2	Recommendation algorithms . . . . .	69
5.3.3	Quality metric . . . . .	71
5.3.4	Algorithms tuning and configuration . . . . .	72
5.4	Results . . . . .	73
5.4.1	Aggregate precision . . . . .	73
5.4.2	Purpose-specific precision . . . . .	75
5.5	Discussion and Conclusion . . . . .	78



<b>6</b>	<b>Designing Recommendations for Mobile Devices</b>	<b>81</b>
6.1	Introduction . . . . .	81
6.2	Rating on Mobile Devices . . . . .	83
6.2.1	Data collection . . . . .	84
6.2.2	Rating efficiency . . . . .	86
6.2.3	Questionnaire . . . . .	88
6.2.4	Discussion . . . . .	90
6.3	Personalized Recommender Algorithms . . . . .	91
6.3.1	Offline evaluation . . . . .	92
6.3.2	User evaluation . . . . .	96
6.3.3	Discussion . . . . .	101
6.4	Comparison with Commercial Services . . . . .	102
6.4.1	Offline evaluation . . . . .	102
6.4.2	User evaluation . . . . .	104
6.4.3	Discussion . . . . .	106
6.5	Learnings and Limitations . . . . .	108
<b>7</b>	<b>Collecting the Initial Ratings from the Crowd</b>	<b>111</b>
7.1	Introduction . . . . .	111
7.2	Problem Definition . . . . .	115
7.3	Framework . . . . .	116
7.3.1	Worker clustering . . . . .	118
7.3.2	Profile computation . . . . .	121
7.3.3	Utility optimization . . . . .	124
7.3.4	Cheaters identification . . . . .	125
7.4	Evaluation . . . . .	126
7.4.1	Rating quality experiments . . . . .	127
7.4.2	Parameter tuning . . . . .	131
7.4.3	Utility function experiments . . . . .	135

7.4.4	Malign workers experiments . . . . .	137
7.5	Conclusion . . . . .	140
<b>8</b>	<b>Planfree - a Restaurant Recommender Service</b>	<b>141</b>
8.1	Introduction . . . . .	141
8.2	Satisfying the Requirements of Recommender Systems . .	142
8.2.1	Learning user's tastes . . . . .	142
8.2.2	Making recommendations . . . . .	145
8.2.3	Availability everywhere and anytime . . . . .	147
8.2.4	Getting interest of users and restaurateurs . . . . .	149
8.3	Architecture and Implementation . . . . .	152
8.3.1	Data layer . . . . .	153
8.3.2	Application layer . . . . .	154
8.3.3	Presentation layer . . . . .	156
8.3.4	Implementation and code base . . . . .	157
8.4	Tests . . . . .	158
8.4.1	Scalability test . . . . .	158
8.4.2	Usability test . . . . .	159
8.5	Research Exploitation . . . . .	161
8.6	Conclusion . . . . .	164
<b>9</b>	<b>Conclusion</b>	<b>167</b>
9.1	Contributions . . . . .	167
9.2	Lessons Learned . . . . .	168
9.3	Limitations . . . . .	171
9.4	Future Work . . . . .	173
	<b>Bibliography</b>	<b>177</b>

# Chapter 1

## Introduction

Leisure time is considered very important nowadays and everyone wants to get the most out of it, without wasting a second of it. People can choose between many leisure activities and anything that make people relax, have fun or forget all the problems of work can be considered a leisure activity. In this thesis, we are considering those activities that can be performed outside our own house, possibly together with friends or the partner. For example, people can choose between dining at a restaurant, watching a movie at a cinema, going to a pub for a beer, dancing or playing soccer at the public park with friends.

Once the activity has been selected, we are not ready yet to get fun: we have to identify a place where we can perform it. A place for leisure activities can be a business that offers services for leisure activities, such as pubs, restaurants or museums, but also a public space that everyone can use, like a public park. Suppose we want to go out for dinner, which restaurant is the best for us? It depends on our tastes, our mood and the type of experience we want to have. We can easily choose which place is the closest to our needs between the ones in which we have already been, but what about all the other opportunities our city is offering? Do we know all of them? Usually the answer to this question is no.

Nowadays cities offer numerous opportunities for leisure time. Having opportunities is great because we have a higher chance to find something that better fits our taste and needs of the moment, but raises some difficulties: i) it is difficult to know all the opportunities available and ii) it is hard to identify the most interesting opportunity between the available ones.

Despite the work presented in this thesis can be adapted to any leisure activity, for the sake of simplicity and clarity, we decided to focus only on one specific activity: going out for lunch or dinner. In this way, we consider only restaurants as what we generically called “places”.

Before the advent of computers and of the Internet, when people had some leisure time to spend, they had some opportunities to find where they could perform some activities.

- *Go around and personally check what the neighborhood offers.* This means that people enter a restaurant and try it, without any planning in advance. They can be lucky and get a fantastic experience, but they could also enter a restaurant that does not correspond to their needs. In this way people get to directly know the opportunities available in the selected neighborhood, but at the cost of wasting their time if the place is not good for them.
- Through *advertisement (posters, flyers, radio or television)*, restaurants show their offerings to people, trying to attract them there. People can get some hints on the type of experience they can get there, but they get to know only about some specific places, while the most interesting one may be hidden somewhere else.
- Looking at *yellow pages*, people can see an almost complete list of what is available around them, but they have only little information about what these restaurants can offer and the type of experience

---

they can have there. Yellow pages contains the addresses and contact information of many and many restaurants, making hard and time-consuming for people to find the best opportunity for their needs.

- *Friends and relatives* know their care ones' tastes and can share with them their experiences, recommending the restaurants in which they can enjoy their leisure time. This is perfect as people do not need to spend time going around in the surroundings or experiencing unknown restaurants: they just go to the recommended restaurant and have high chance to have a good experience there. On the other hand, friends and relatives have only partial knowledge of the available opportunities in their city as they can speak only about the restaurants they had the possibility to experience.

With the advent of the Internet, new possibilities became available. Maps and yellow pages-like services let people find an almost complete list of the businesses, making the search for restaurants close by faster than the paper-based yellow pages. But still, the overwhelming amount of information make it difficult to find the opportunities that are interesting and people need more support to easily and quickly identify a restaurant where they can get the most out of their leisure time.

For these reasons, online recommender services have been developed: they collect comments and opinions from people that experienced the restaurants available in the city and report them to other people. In this way, people can learn from others' experiences the type and quality of experience they can have in a restaurant and can relate it to their tastes to decide whether the restaurant is interesting to them or not. Such generic services recommend the restaurants that are liked the most in general, making people spend some time to read the reviews, understand the restaurant

characteristics and quality, and relate it to their tastes before being able to take a decision.

Personalization makes a step further: personalized recommender services learn people tastes through their reviews to known restaurants and recommend them the unknown restaurants they should like.

The objective of this thesis is to study how personalized recommendations can be applied in the context of restaurant discovery. We start by understanding how people choose a restaurant, identifying the information that influences their choices. The same information needs to be considered by the recommender service to better tailor the results to the specific situation of the requester. We model how these data can be collected and used to improve the quality of personalized recommendations and we demonstrate the applicability of our results by implementing them in a restaurant recommender service.

## 1.1 Requirements

Personalized recommendation services solve the problem of overloading of information that people feel when they look for a place where they can spend their leisure time. Despite some of these services are already available, still friends and relatives have a deeper understanding of people's tastes and can easily adapt to specific requests people do, related to their context, their mood and the way in which they prefer to spend their time in that specific occasion [70].

To design a personalized recommender service, we need to tackle different problems. First, it is fundamental to collect people's opinions about the places in which they spend their leisure time and to *learn from them their tastes*. Second, we need to understand how to use this information to *identify the best place for a specific requester*, considering her specific

needs at the moment of the request. Data collection and recommendation building are the most important elements of a recommender system, but *the service needs some data to start working*: a complete knowledge of the leisure-time offerings of the area and an initial set of people's opinions that can be used to understand how good the places are, to make the recommender algorithm able to build useful recommendations even for the first user coming in. Finally, people search for leisure places *everywhere and anytime*, so the service should be always available, providing users the needed information without the need to plan their leisure time in advance.

In the following, we look into the details of these different aspects of a recommender system and identify the requirements that a recommender service should satisfy.

### 1.1.1 Learning user tastes

The most important information used by a recommender system is people's opinion about the items it wants to recommend. Such opinions can be collected in many different ways. The system can collect *implicit feedback* by tracking how people interact with the items (visualization, purchase, ...) or can ask for *explicit feedback* in form of rating, review or detailed description of the past experience with the item.

The implicit feedback has the advantage of being hidden from the user, resulting in this way effortless. The information collected is in the form of a history of what the users searched for, what attracted their interest and which items they actually needed. The amount of data that can be collected in this way is huge, and these data need to be interpreted and reduced to a shorter representation of users' tastes, used then to identify which other items could be of interest for them the next time they access the service.

Explicit feedback, on the other hand, requires a direct involvement of users, requiring them to spend some time to express their opinion. Such effort can be low or high, depending on the amount of information the system wants to collect. For each item the users know, the system can ask for different types of explicit feedback.

A rating is just an indication of how much the user liked the specific item and is expressed by selecting a value from a rating scale. Many rating scales are available, each one with its pros and cons, different levels of effort needed to select the value representing users' opinion and different granularity. Through ratings the system can learn how much the user liked the item in general, while different characteristics can be learned with multi-criteria ratings, in which the user can express her judgement for the listed characteristics of the item (like service, food quality and ambience in the case of a restaurant). Ratings are usually converted into numeric values that are easy to manage from a computational point of view and the recommender algorithms can easily find which items are preferred.

Textual reviews let users express their feedback with more freedom, letting them write anything they think that could help other people understand the characteristics and quality of the item.

Finally, a service could collect explicit feedback through a series of questions that can ask for a rating, a review and extra details about the experience the user had with the item, such as when she experienced it, where, with whom and how she judges it according to different aspects. Such a complex explicit feedback requires much effort to the user, but provides the system many details about the experience of the user with the item that can be used to understand both user tastes and item characteristics.

Despite textual reviews and specific questions could let us collect a huge amount of information about each single restaurant, we decided to limit



our analysis of people's feedback to ratings: they are quick to insert and requires less effort to people.

### 1.1.2 Identifying the best restaurant

Once user feedback has been obtained, we need to identify the best algorithm that learns the most information possible from these data and computes the best recommendations possible.

Once user tastes are learned, we need to understand the needs of the requester at the moment in which recommendations are requested. Different contexts could require different recommendations according to time, requester's location, mood and, possibly, companions. While these context information are useless in e-commerce recommenders (as products can be send everywhere and anytime), they become important when recommending for leisure activities. For example, if a person is in Rome, Italy, and is searching for a place where she can have dinner, she does not like to receive recommendations about restaurants in a different city. If she is really hungry and can move only by foot, she wants only recommendations about restaurants close by, which she can reach easily by walking few minutes. Moreover, the requester does not want to reach the recommended restaurant and find it closed, so opening hours of the places and time of the request should be considered.

There is much context information that could be useful to better tailor the recommendations, but we have to consider both its usefulness and the effort needed to collect it. Data like location and time can be implicitly collected with the information about the received request, but other details about companions or mood could need a direct involvement of the requester. Then, *the recommender algorithm needs to include such context details to improve its results.*

### 1.1.3 Availability everywhere and anytime

We have seen that before the use of computers and of the Internet, people were able to search for place for leisure activities through yellow pages. This solution was available only at home, where the book was kept, and it was time consuming to check all the available opportunities and find one in the selected area that was satisfying the users needs. This required to plan in advance what to do during leisure time. Even friends and relatives recommendations are not always available.

Thanks to the Internet it is easier to access to more information, and with mobile devices we can receive recommendations everywhere and anytime. Making a recommender system available also from mobile adds extra requirements.

Suppose we are in the city center, it is dinner time, we are hungry and we want to find a restaurant close by where we can have fun with our friends. We take our smartphone and access the recommender service. If we are not first-time users, the system recognizes us and already knows our tastes, so we would like to immediately get recommendations after just choosing the type of activity we are interested in. This means that the service should *automatically detect most of the context*: it should recognize where we are and that it is dinner time, meaning that we are interested in restaurants (the type of activity could be specified by us) that i) are close by, ii) can be reached within few minutes and iii) are open right now.

Given the context has been learned without any effort from the user, the system should be able to compute the recommendations *very quickly* and the user should immediately find the most interesting option: current services show recommendation lists of tens of items, while a mobile user could not have time to scroll it. The recommendations should be *very*

*precise* and the user should find the most interesting option in the first few positions.

#### 1.1.4 Starting the service

Once we know how to collect user feedback about items and we know how to use it to build recommendations, we have to collect the initial data needed to start the service. Let's suppose we start from a selected city.

First, we need the complete list of the *places for leisure activities available* in the city. These data can be added manually by hiring people that know well the area and that are willing to explore the city and to insert every place they see. This solution is quite costly and time-expensive as there are many places for leisure activities even in a single neighborhood. An alternative is to use services like Google Maps (<https://www.google.it/maps>) or Open Street Maps (<https://www.openstreetmap.org/>), where maps of points of interests can be accessed and the categories of places we are interested in can be retrieved and used through these services. The cons of using these services is that they could provide incomplete or erroneous data as businesses change.

Once we have all the items we are interested in, the recommender algorithm needs an *initial set of people's feedbacks* to be ready to build recommendations as the first users arrive. When the algorithm receives the request for a recommendation list, it builds its predictions using the feedback shared by previous users, according to a logic that is specific for the algorithm. This means that if the first user makes a request and no one already shared her opinion about the known items, the algorithm does not know anything about the items and have no idea about which one would be more interesting for the requester, and no recommendations can be built.

Another challenging situation is when a new user accesses the service: we do not know her tastes yet and we need a way to learn them quickly.

One solution could be to force the new users to give some feedback before accessing the recommendation feature. In this way we can build personalized recommendations even at the first request, but we have to design this collection of feedback in a way that it does not discourage people to try the service. Another option is to build the first recommendation list as generic, without any hints on the tastes of the requester, explaining the user that if she adds feedback to the places she knows the recommendation quality will increase. The risk of adopting this solution is that the user receives recommendations of bad quality and decides to not use the service any more or she never adds her feedback, without giving any contribution to the service she is using. The main problem here is to maintain a balance between the consumption of and contribution to the service of each user, as both actions are fundamental to keep the service alive.

## 1.2 Methodology

We have seen that recommender systems have many different challenges to take in consideration.

To tackle them we started by analysing the *state of the art* on the related topics:

- Discovering places, and in specific restaurants and similar businesses. We identified the services that support people discovery of new restaurants and are currently used, analysed them and spotted out pros and cons of using them.
- Recommending items. Many recommendation algorithms have been already developed in the last years. We explore them and identify their strengths and weaknesses, their adoption in recommending places and restaurants, and their integration with social networks.

- Expressing ratings. Different rating scales have been proposed and used in different services. We present them and discuss how they influence people’s feedback. Moreover, the problem of collecting ratings is analysed and the research work about crowdsourcing such subjective data is summarized.

We started our research work by *understanding the role of friends*. Nowadays, social networks let us easily collect friendships between people. Since leisure activities are usually performed with some companions, we studied whether friends, as stated in social networks’ relationships, can have an effect on users’ opinions, or whether friends’ opinions influence people’s ones. For this reason, we collected a dataset of restaurant ratings and friendships as stated on Facebook and analysed the effects of different user-bases on collaborative filtering recommendations.

We continued by exploring *the effect of purpose*, i.e. is the reason why people are willing to go to the restaurant influencing their choice of the place? We identified three companion-based purposes and a cost-related one: having dinner with tourists, with the partner or with friends and lunch break (meaning good price/quality ratio). For each restaurant, we collected people’s feedback for each purpose identified and evaluated the effect of these purpose-based ratings on restaurant ranking and recommendations. To quantify our results we compared the obtained recommendations with the ones provided by a popular restaurant recommender service: TripAdvisor.

Once learned how to collect people’s feedback and which extra information boosts the quality of recommenders’ input data (i.e. friendships and purpose-orientation), we run a *comparative analysis of possible solutions for collecting ratings and recommending restaurants*. Different rating scales and recommendation algorithms have been both studied through objective metrics (such as precision and recall) and evaluated by users, identifying

the combination of rating scale and recommender algorithm that provides the results of higher quality and that better satisfies users.

Before being able to compute recommendations, a recommender service needs an initial set of ratings from which it can learn which restaurants are good and which are not. We designed a *novel crowdsourcing platform for collecting reliable ratings*. A reliable rating is a truthful rating from a worker that is knowledgeable enough about the item she is rating. Through this platform, restaurants can receive ratings of high quality as lazy and malign workers are identified through a skill-based system which respects the subjectivity of the expressed opinions. In addition, our platform focuses on acquiring ratings for items that only have a few ratings. We evaluate the proposed framework through simulations on synthetic and real datasets, measuring the quality of the reliable ratings by using them as dataset for a recommender service.

Finally, we collected all the findings and used them as a guide for the implementation of a novel restaurant recommender service. Some extra adjustments were needed to adapt the recommender service to an online service, where scalability and implementation issues have to be considered.

### 1.3 Contributions and Results

Our work produced the following contributions.

*Friends with similar tastes are the best user-base for collaborative filtering.* Even the simpler user-based collaborative filtering recommender can be improved if it selects neighbors as the friends with tastes closer to the requester. Surprisingly, this works even with friendships as stated in social networks, which usually contain relationships of different kinds and strengths [80], and not only the friends we spend our leisure time with.

*People choose different restaurants according to their purpose*, i.e. according to the type of companions they have: going out with tourists is different than going out with the partner or with friends. Moreover, when the price/quality ratio is taken into consideration, for example as when choosing a restaurant for lunch break, the opinion of people changes.

*Similarities in the preferences of people can be extended to other activities*, which points to the potential of profiling users based on lifestyle. Moreover, the effect of purpose is confirmed for other leisure activities, such as having a beer at a pub.

*Locals are more knowledgeable*. We found that locals, i.e. city dwellers, have a better understanding of the qualities of the different offerings in the city: they have the opportunity to experience many of them and they can compare them. By collecting purpose-based evaluations of restaurants from locals, we obtain a good-quality dataset that let even off-the-shelf recommender algorithms build good-quality recommendations, which easily outperform, for instance, TripAdvisor.

*5-star ratings and user-based collaborative filtering provide the best recommendations, both according to precision and user satisfaction*. 5-star rating scale is preferred over smaller scales even when providing ratings through mobile devices and in situations of split attention (i.e. while walking). Between collaborative filtering algorithms, the user-based one is able to provide recommendations of higher quality when providing short recommendation lists (with only 5 restaurants). Recommendations computed with this combination of rating scale and algorithm perform better than TripAdvisor and produce recommendations of quality comparable to Foursquare.

*Skill-based evaluation of workers can detect cheaters, such as lazy and malign workers, in a crowdsourcing platform for collecting restaurant ratings*. Lazy workers, i.e. cheaters assigning ratings randomly, are detected

with 70% precision and 80% recall, meaning that only 20% of them is able to hide their random behaviour. Moreover, malign workers are identified with almost 90% of precision when at least 20% of their ratings is misleading, i.e. aimed at raising or lowering the average rating of some items.

*Reliable ratings provide high-quality recommendations.* Evaluations of the proposed crowdsourcing platform confirm that removing ratings from workers suspected of being cheaters the quality of the recommendations obtained is raised by more than 50%.

As part of our studies, we collected four *datasets of purpose-based ratings*, one for restaurants, bars, pubs and clubs in 3 different cities around the world and three only for Trento’s restaurants. Such datasets are fundamental for studying recommendation algorithms and let researchers test algorithms in these peculiar conditions. To the best of our knowledge, there are not other datasets with similar characteristics.

## 1.4 Structure of the Thesis

### Chapter 2. State of the Art

In this chapter we analyse the past research work about the topics related to this thesis: the discovery of places, recommender algorithms (with a focus on the services that recommend restaurants or related to the leisure sector), and rating collection. These topics have been studied under different points of view, and we will show which research questions remain unsolved. The chapter is based on and extends the State of the Art deliverable presented in the context of the project Toolisse [75].

### Chapter 3. The Role of Friends in Decision Making

We explore the importance of friend recommendations. We present an experiment in which we mixed social network-based relationships with collaborative filtering recommendations to verify how this kind of “friends”



reflect our tastes and whether they have potential to make us discover new places we actually are interested in. These results have been extracted from the paper published at CollaborateCom [70].

#### **Chapter 4. How Purpose Influences Opinion**

We study the factors that affect people’s decision in participating to leisure activities. To this end, we collected the ratings of local people from three different cities around the world on popular leisure activities, and looked at the personal, social and contextual features shaping their preferences. We then used this dataset to evaluate how these features can be exploited to recommend places people would actually like. This chapter is an extension of the paper published at CGC [71].

#### **Chapter 5. Purpose-orientation and Focus on Locals**

Contrary to most works on the topic, in this chapter we do not focus on the algorithmic side of the problem of recommendation quality and instead study the importance of the data in input to the algorithms. We study the case of restaurant recommendations for locals and compare the effect of purpose-based and local-provided ratings on recommendations as opposed to generic tourist-based recommendations provided by services like TripAdvisor. These results have been submitted at Internet Computing [73].

#### **Chapter 6. Designing Recommendations for Mobile Devices**

In the context of leisure activities, recommender systems can merge the word-of-mouth of friends and relatives with a complete knowledge of the available options. Making this service reachable from mobile devices can provide recommendations anytime and everywhere, avoiding the need for planning in advance. In this chapter, we focus on restaurants, we study how to collect ratings on mobile devices and which algorithm is best for mobile restaurant recommender services. This chapter is under review process for Software: Practice and Experience [74].

### **Chapter 7. Collecting the Initial Ratings from the Crowd**

In this chapter we address the problem of acquiring reliable ratings of items such as restaurants from the crowd. We propose a platform that recognizes lazy and malign workers through the identification of their skills as they insert more ratings. Moreover, the items to be rated are assigned in a way that items with less ratings are exposed more and, in particular, to workers with higher skills for them. This research work has been submitted to WEBIST [76].

### **Chapter 8. Planfree - a Restaurant Recommender Service**

In previous chapters we presented our studies identifying the best solutions to the challenges for recommendation systems for leisure activities. In this chapter we see how these learnings can be applied to a restaurant recommender system. This chapter is an extension of a technical report related to the Toolisse project [72].

### **Chapter 9. Conclusion**

Final discussions about the presented research work, its limitations and future work are presented in the concluding chapter.

## **1.5 List of Publications**

- Beatrice Valeri, Fabio Casati, Marcos Baez, and Robin Boast. A model for knowledge elicitation, organization and distribution in the cultural entertainment sector. In *Human Centric Technology and Service in Smart Space*, 2012.
- Beatrice Valeri, Marcos Baez, and Fabio Casati. Comealong: Empowering experience-sharing through social networks. In *International Conference on Collaborative Computing: Networking, Applications and Worksharing (Collaborate-Com)*, 2012.

- Galena Kostoska, Denise Fezzi, Beatrice Valeri, Marcos Baez, Fabio Casati, Samuela Caliarì, and Stefania Tarter. Collecting memories of the museum experience. In *CHI13 Extended Abstracts on Human Factors in Computing Systems*, 2013.
- Galena Kostoska, Denise Fezzi, Beatrice Valeri, Marcos Baez, Fabio Casati, Samuela Caliarì, and Stefania Tarter. Understanding sharing habits in museum visits: A pilot study. *Proc. Museums and the Web*, 2013.
- Beatrice Valeri, Marcos Baez, and Fabio Casati. Come along: understanding and motivating participation to social leisure activities. In *International Conference on Cloud and Green Computing (CGC)*, 2013.
- Beatrice Valeri, Marcos Baez, and Fabio Casati. A tool for motivating participation to cultural events. In *CHIItaly*, 2013.
- Galena Kostoska, Denise Fezzi, Beatrice Valeri, Marcos Baez, and Fabio Casati. Sharing museum experiences: an approach adapted for older and cognitively impaired adults. In *Proc. Museums and the Web Asia*, 2013.
- Beatrice Valeri, Florian Daniel, Fabio Casati, Mafè de Baggis, and Filippo Pretolani. D5.1 Recommendation and Emotional Representation of Places and Events: State of the Art. Technical report, TrentoRise, 2014.
- Beatrice Valeri, Florian Daniel, and Fabio Casati. D5.3 Planfree - Concepts, Architecture, Implementation and Exploitation. Technical report, TrentoRise, 2015.
- Beatrice Valeri, Florian Daniel, and Fabio Casati. Better recommendations of social leisure activities through better understanding of peo-

ples choices and motivations (poster). In *International Conference of Computational Social Science*, 2015.

- Beatrice Valeri, Shady Elbassuoni, and Sihem Amer-Yahia. Acquiring reliable ratings from the crowd (poster). In *Conference on Human Computation & Crowdsourcing (HCOMP)*, 2015.
- Beatrice Valeri, Florian Daniel, and Fabio Casati. On the value of purpose-orientation and focus on locals in recommending leisure activities (submitted). In *Internet Computing*, 2016.
- Beatrice Valeri, Florian Daniel, and Fabio Casati. Rating scales and algorithms for mobile recommender systems: The case of restaurant recommendations (submitted). In *Software: Practice and Experience*, 2016.
- Beatrice Valeri, Shady Elbassuoni, and Sihem Amer-Yahia. Crowdsourcing reliable ratings for underexposed items (submitted). In *International Conference on Web Information Systems and Technologies (WEBIST)*, 2016.

# Chapter 2

## State of the Art

In this thesis we are going to tackle many problems related to recommender systems. In this chapter we will explore the solutions already available. We start from the discovery of places, how people perform this activity and which services are already available to support them. Then, we proceed by analyzing recommender algorithms, how they are evaluated and applied to the discovery of places for leisure activities, and how collaborative filtering is affected by social networks. Ratings are the heart of any recommender service: the different rating scales are analysed and crowdsourcing platforms are presented as a means for collecting crowd opinions when volunteering is not enough. Finally, the chapter is concluded with considerations about the questions that are still open and the areas in which further research can provide some improvement.

### 2.1 Discovering Places

The Web has changed the way people discover things to do in their spare time, for dinner, for fun, for work. It is the world's largest knowledge base, and it allows them to look up, find and inspect the description of places and events within seconds. Yet, choosing is in general not as simple as it might sound. The set of search results is typically large, descriptions are not

always satisfying, sometimes they are not trusted or even objectionable, friends must be consulted as well, etc. In the following, we overview that state of the art of how people discover places on the Web with the help of dedicated applications or services and of how they share their opinions or participation with their friends, giving their experience also a social dimension.

Over the Internet, many services allow people to discover new places for their leisure time exploiting the knowledge of the crowd, such as real time location-sharing services, partially crowdsourced travel guides and social networks just to mention some.

*Location sharing services* are very popular: users can see where their friends are in a specific moment and can make serendipitous meetings thanks to such information. Swarm by Foursquare ([www.swarmapp.com](http://www.swarmapp.com)) makes it easy to discover new places nearby, added by the users themselves when they decide to share their position (i.e. they check-in in a place). The location sharing is motivated both by discounts offered by the place owner, where available, and by a game: the user collects some points each time she shares a location, earns badges and becomes the major of a place. Swarm is also a social network, where users can select their friends and receive notifications about their movements, keeping updated with their location.

Other than the explicit motivations offered by the system, i.e. rewards, there are other reasons for location sharing, such as life-logging, communication and coordination with companions, recommendation of a place to friends [11]. Swarm is used also for bookmarking places or let friends know that they are available or that they reached their destination safely, in particular when they check-in to their house. These behaviours have been identified in [44]. The authors studied the effects of points and badges, and discovered that they are good motivators for beginners, while later users continue to use the service because of the social network connected to it.

Moreover, an important motivation to not check-in was identified in self-representation: people do not check-in into places that could be considered bad or that reveal a bad behaviour according to their friends [44].

The usage of location-sharing services has been studied also with the goal of designing models to predict users' future check-ins [4, 8, 53]. Knowing the user previous check-ins and the check-ins of her friends, Chang and Sun [8] were able to recommend to the user her next checkin with 90% of precision. Moreover, they found that if two people often check-in into the same places, they are likely to be friends.

With location-sharing services people can discover new places in their own city and in the surroundings, knowing where their friends go. For tourists visiting a new city such services are not so helpful and *travel websites* (or portals) are typically more focused on their context and needs. Tourists find very useful the services based on users' feedback that recommend hotels, restaurants and places to visit, such as TripAdvisor ([www.tripadvisor.com](http://www.tripadvisor.com)) [49, 32]. Ratings coming from business customers are trusted more than expert ones: experts' opinions can be biased because of personal profit. Recommender systems' ratings collect people opinions without any control on their knowledge of the place they are rating, letting users free to add also fake reviews to increase the ranking of their own business or to lower the one of their competitors [18]. Also some TripAdvisor's reviews seem fake, but the number of honest reviews is so high that the fake ones do not have any impact on the final ranking of businesses and the system is still reliable [54]. At our knowledge, user's satisfaction of TripAdvisor recommendations has not been studied yet.

Gogobot ([www.gogobot.com](http://www.gogobot.com)) is another recommender system focused on tourists, posing itself as a competitor of TripAdvisor. Differently from TripAdvisor, it lets people ask for a partially-personalized recommendation list. Gogobot users build their profile entering in some communities

already provided by the platform, each one describing a lifestyle aspect. Putting in your profile each community that represents one aspect of yourself, you let the system know about your interests and the personalized recommendation lists are more likely made of places that are interesting for you. Gogobot's knowledge about places comes from the crowd, and each review contains a rating for each community that is well represented by the place.

Some other services are similar to yellow pages, such as Google+ Local ([www.google.com/local](http://www.google.com/local)) and Yelp ([www.yelp.com](http://www.yelp.com)). Both these services help people finding businesses and places they are interested in. They collect people's feedback in form of 5-star ratings and textual comments.

Other services are more focused on restaurants and places where also locals go. Zagat ([www.zagat.com](http://www.zagat.com)) is a trusted recommender system for restaurants, hotels and nightspots, based on multiple 30-scale ratings for each place, each one on a different characteristic. This rating system allows more refined recommendations and a deeper understanding of the type of experience that the place can offer. Moreover, Zagat collects ratings from experts and their own editorial team curates all reviews and recommendations. Experts are kept in the loop also by Urbanspoon ([www.urbanspoon.com](http://www.urbanspoon.com)): experts can be biased, but they still have a better understanding of all the available options and they can integrate the information already collected by customers. In 2014 Foursquare added local search functionality to its application ([blog.sweetiq.com/2014/08/foursquare-new-improved-yelp](http://blog.sweetiq.com/2014/08/foursquare-new-improved-yelp)), using both the check-in information of the location-based social network and user feedbacks, reviews and tastes to provide recommendations. Foodspotting ([www.foodspotting.com](http://www.foodspotting.com)) moves restaurant recommendations at a deeper level, going directly to the food served in each place and building user profiles with food tastes. Actually, this is a very important information that is very useful when you have to



decide which restaurant to go to. The Fork ([www.thefork.com](http://www.thefork.com)) is a restaurant booking service, which includes also recommendation functionalities. Here only people that booked a table in the restaurant can rate it: in this way the feedback is collected only from people that actually have been in that restaurant.

Although recommender systems have been studied for a while, their application in the context of leisure activities still needs more understanding and the development of new applications like Foursquare local search and The Fork is a clear indicator of the activity and evolution in this area.

## 2.2 Recommending Items

With the Internet, an overwhelming amount of information became available to everyone, but finding the interesting content became more and more difficult. Recommender systems were born to solve this problem: among a huge amount of items, they search the more interesting one for the user, when she does not know the exact item she is looking for. These systems are based on user's opinions about the available items, collected mostly in form of rating.

Recommender systems can be divided in two main categories: content-based recommender systems and collaborative filtering. *Content-based recommender systems* [46] first learn the user profile, by extracting the characteristics of the items the user liked, then build recommendations putting at the top of the list the items that have more of the characteristics present in the user profile. These recommenders have three main advantages: i) each user is independent of the others, because recommendations are determined only by the user's own ratings; ii) the recommendations are transparent and the characteristics of each item can be shown, allowing the user to verify how much it corresponds to her profile; iii) new items can be recom-

mended as well, even though they have not received ratings yet, because the recommendations are based on their characteristics. On the other hand, item characteristics are difficult to extrapolate and are very specific for the context. Moreover, new users have to rate a minimum amount of items before the system is able to build a valuable profile.

*Collaborative filtering* [37], instead, is able to build recommendations without knowing item characteristics, as it is often the case. In [64], collaborative filtering is seen as an “automated word of mouth”, since recommendations are built using other users’ opinions. The user profile consists of her own past ratings assigned to items and is used to determine the taste similarity between users. Items liked by more similar users are then recommended to the requester. In this way, recommendations are computed without knowing extra information about the items and are more varied since users can like items with different characteristics.

In the context of leisure activities and places, some item characteristics need to be considered, like location, time and user’s distance, so we have to filter “recommendable” items according to these characteristics. In this way we restrict the item list to only places the user can actually reach, but we still have a long list full of non-interesting items. At this point, we can apply a collaborative-filtering recommender, to put at the top the places that the user could find more interesting.

### **2.2.1 Main collaborative-filtering strategies**

All collaborative filtering algorithms start from the rating matrix. It has items as columns, users as rows and each cell  $(i,j)$  contains the rating that the user  $i$  assigned to the item  $j$ . The ratings, as we will see more in details in Section 2.3.1, can usually be unary (where the rating is defined, it means that the user likes the item), binary (like vs. don’t like) or numerical, in a finite scale (each number in the scale define the different degrees of liking).

Collaborative filtering strategies can be divided in three main categories: memory-based, model-based and hybrid, which mix memory- and model-based strategies. At each request of a recommendation list, *memory-based strategies* uses the full rating matrix. *Item-based strategies* compute similarity between items and recommend the items more similar to the ones the user liked [61]. The similarity between two items is computed considering how different the ratings given to the considered items by each user that rated both are. Similarly, *user-based strategies* [15] compute the similarity between users and the requester, select the more similar ones (building in this way a neighborhood) and recommend the items that these similar users liked. In this case, similarity is computed comparing how different the ratings given by the two users on the items that both rated are. There are different similarity metrics and strategies to identify the similar users, that have impact on the algorithm's performances. Memory-based algorithms have some problems: i) the ratings matrix is sparse and finding similarities can be difficult; ii) the cold-start problems refer to the inability to compute recommendations for users with no or few ratings, or to recommend items that have not been rated.

*Model-based recommenders*, as the name says, use the rating matrix to build a model, representing the inferred knowledge in a more compact way. This model is computed once and used to answer all the next requests, but needs to be recomputed periodically as new ratings are added. Singular value decomposition (SVD) [37] is the most famous model-based algorithm. SVD decomposes the rating matrix in three smaller matrices according to math-computed features: instead of storing each single rating, a small set of features is computed according to the known ratings. A user-feature and a feature-item matrices are computed, plus a feature-feature matrix that translates computed ratings from one feature to the other. Each computed rating in these matrices represents a piece of knowledge that previously

was represented by many ratings. With a fast vector product, the final prediction for a user and a new item is computed. The quality of SVD recommendations depends on the number of features and the number of iterations run to extract them. Slope One [41] is another model-based algorithm based on a matrix of rating differences between items. The algorithm uses item differences and requester’s actual ratings to compute a predicted rating for the new items, which are then ranked and the top ones are recommended. Model-based algorithms are usually more complex and have long model-building phases, that have to be carefully planned to maintain the recommendations good.

These two types of collaborative filtering strategies have been merged in some *hybrid algorithms*, trying to get the best out of both. *Cluster-based recommender* [79] “models” item and user neighborhoods, grouping them in clusters according to their similarity. When a recommendation request arrives, the rating matrix generated with the ratings of users in the requester’s cluster is used to apply a memory-based recommender algorithm. The clustering algorithm, the similarity metric and the number of clusters computed are the parameters that affect the quality of the recommendations.

### 2.2.2 Algorithms evaluation

In the past years, many researchers have tested the different collaborative filtering recommender algorithms, trying to understand which one gives the best recommendations. Researchers used different formulae to understand how good the recommendations are, considering both precision of top-k recommendations and errors in singular rating predictions [7, 40]. In [7] the authors compared 11 different collaborative filtering algorithms (and some variants of one of them) on MovieLens and Netflix datasets, which contain movie ratings in a 5-star scale. The tested algorithms are: user-

based, item-based, similarity fusion, regression-based, slope one, LSI/SVD, regularized SVD, integrated neighbor-based - SVD model, cluster-based, personality diagnosis and tendency-based algorithm. All these algorithms have been tested both for prediction precision (how close is the predicted rating to the actual rating) and for top-k recommendations (i.e. building the recommendation list). The algorithms were tested under different data densities, computing all the most common evaluation metrics: coverage, MAE (Mean Absolute Error), RMSE (Root Mean Squared Error), Precision, Recall, ROC (Receiver Operating Characteristics) and Half-life test. Testing algorithms with different densities, the authors found that the recommendation quality of model-based algorithms changes less as density changes, but on the other hand, memory-based algorithms are more accurate when data density is high, because model-based algorithms abstract and collapse data losing part of their knowledge. The best results were obtained with SVD-based algorithms, followed by tendencies-based and slope one. Cluster-based algorithm performed the worst: it is based on the existence of different user communities, but apparently in the used datasets such communities are not present.

Lee, Sun and Lebanon [40] performed a similar test of collaborative filtering algorithms (with Netflix dataset), coming to the same conclusion that regularized SVD is the most accurate recommendation algorithm between the 15 selected. They tested the algorithms also considering time constraints and found that regularized SVD is slow in building recommendations, taking at least 5 minutes under any density condition. For real-time recommendations the authors recommend to use simple user average, while the best algorithm that can build recommendations in maximum 1 minute is Slope One, when the density is at least 2%.

The evaluations based purely on mathematical formulae for precision and error do not always reflect the users' perceived quality of recommen-

dation lists. User-based evaluations can give more information about perceived system qualities, beliefs, attitudes and behavioural intentions, and can provide a more complete feedback on the full recommender system [56]. An example of recommender algorithms evaluation based on users is available in [12], where the authors asked people to evaluate recommendation lists considering perceived accuracy, novelty and satisfaction. The baseline recommender of most popular items and a variant of SVD resulted as the most accurate and generated the highest satisfaction in users, but their novelty was low.

### **2.2.3 Recommending places**

We have seen that recommenders work with generic items and the requester's interests and past experiences are directly related to the future experiences she will have with the recommended items. When we speak of leisure time, we are speaking of social activities that are performed together with friends. When we recommend an activity for leisure time, the user context is really important, differently from the recommendations of e-commerce products in which only item characteristics and user preferences are needed. Baltrunas et al. [5] show that recommendation systems are able to increment user satisfaction by considering weather conditions, companions, time (season, weekday or time of the day) and familiarity with the area, along with other contextual information. Mobile devices provide support for context-aware recommendation systems. Thanks to their sensors, they can automatically collect contextual information, such as user position and therefore weather conditions, and also provide for proactive recommendations [77]. For example, the recommendations for a user that is doing shopping in the city center at lunch time with a friend should involve restaurants that can be reached by foot in few minutes and that are cheap. In other situations restaurant recommendations could use the

context information to identify restaurants with different characteristics or recommendations could be useless.

In [28] the authors found that people mostly rate restaurants in an area of 14 miles (22 kilometers) in diameter: this indicates that restaurants conveniently situated are within this distance from the user location, while farther places are less likely to be interesting to the user. Moreover, people that go to the same restaurants tend to live close to each other. This result is very useful for neighborhood-based collaborative filtering, since neighbors search can be restricted to close users that live nearby, improving efficiency without losing precision.

For tourists, the recommendations are very important because they don't know the area and the available options it offers. Recommendations for tourists do not involve just the activity and place to perform it, but are related also to time and path to reach places since the user is not able to predict them by herself [21]. In tourism, some recommenders also consider multi-criteria ratings to understand better what the user likes of the different aspects of her experience, while other recommenders are critique-based, letting the user guide better the recommender by expressing why an item is not interesting for her and, in this way, implicitly adding more details to the search query [21]. Ricci in [57] went one step further, identifying in the item representation another challenge for recommendation services. Recommenders should make it clear why they recommend an item, giving some information about it. In this way, they let the user understand which type of experience she could have going in that place or performing that activity.

#### **2.2.4 Social networks and collaborative filtering**

Mui and his colleagues [51] already understood in 2001 that each person gives different reputation to others and their opinions should have different

effects on the predicted rating of the user computed by the user-based collaborative filtering algorithm. Other researchers used profile similarity and rating overlap to improve the quality of recommendations [6]. Since offline recommendations are received mainly from friends and familiar advisors, i.e. people that the user knows and trusts, Bonhard and his colleagues [6] proposed to apply collaborative filtering only on a more personal user-base: people that have the same interests (coming from their profiles) or that rated the same items in the same way.

In the last years, thanks to social networks, more people's information became available on the Internet: friendship and trust relationships. Such information can be used to improve the recommendations generated with collaborative filtering [63, 45, 47, 67]. In [63] the authors found that there are two types of relations that, combined together, enhance the quality of recommendations: they called them social friendship and spiritual friendship. *Social friendship* is the actual friendship relation stated by users, while *spiritual friendship* is defined by the similarity in behaviour, i.e. two users have a spiritual friendship when they like the same things. The conclusions in [45] are quite similar: when collaborative filtering is applied on a user-base composed of user's friends and neighbours (i.e. people with similar tastes) the resulting recommendations are better than using only one of the groups.

In [48] the authors present recommender algorithm that mixes trust with friendship and social networks. The authors paid a lot of attention to user's privacy and decided to use local approaches: users can completely manage their profiles and share them on demand to direct friends. When the recommender needs to predict the user's rating (i.e. to assign a score), the ratings of direct friends are asked together with their trust (explicitly assigned by the user). The score is computed weighting friends ratings with their trust values, or it is assigned a default value if there are not enough



information to compute it. If the user’s friend does not have a rating for the item, the system computes the friend’s score (i.e. the predicted rating) and then only the score is shown, without propagating the ratings of friend’s friends. Finally scores are aggregated before showing them. Moreover the authors decided to add other weights, other than the explicit trust, to better represent the value of relations between friends: correlation indicates how similar the friends are, giving higher importance to the ratings coming from friends that have similar tastes, while confidence indicates how the propagated score was computed, allowing the recommender to distinguish between real ratings, computed scores and default score.

In taste related domains, friend relationships can really improve collaborative filtering: in [23] the authors recommended Munich clubs to users in a social network by selecting neighbors between friends. They found that “groups of friends are more similar in ratings of taste related domains than groups of people that do not know each other”. These results were confirmed also in [65], where the authors found that restricting the neighbors search between friends, user-based collaborative filtering maintains the same performances with a reduction of resources consumption.

## 2.3 Expressing Ratings

To build a recommendation system we need three ingredients: i) items, together with their characteristics, to be rated; ii) users willing to share their opinions; iii) ratings, i.e. the representation of users’ opinions. Before the system can start building recommendations, we already have to define all the items we are interested in recommending, and we have to collect some ratings from some users. Without an initial set of ratings, the recommendation algorithms are not able to build the first version of model (for model-based collaborative filtering algorithms) or is not able to compute



Figure 2.1: Examples of different rating types.

similarities between items or users (for memory-based collaborative filtering algorithms). This leads us to two main problems: i) how we represent users' opinions, i.e. in which form ratings are collected; ii) how we find people to give the first set of ratings.

### 2.3.1 Rating representation

People can express their opinions in many different ways: ratings can be unary (only “I like it”), binary (thumbs-up / thumbs-down), numerical in a small finite scale (such as 5-star ratings), in larger scales or continuous in a fixed interval (such as Jester’s ratings).

*Unary ratings* are the ones that only identify whether a person likes an item. We can see an example of such ratings in Facebook, when users are invited to say “I like it” to pages, movies, tv programs and others, but also bookmarks can be seen as unary ratings. This type of ratings is particular because the system does not collect any information about “bad items” and the similarity between users, where needed, should be computed in a different way.

The most known *binary ratings* are the ones of YouTube. They let people express both positive and negative opinions, but without any degree. The recommender system is not able to distinguish a good item from an awesome one.

The other numerical rating scales let users express different shades of their positive, negative or neutral opinion, at different granularities. *5-star ratings* are quite a popular example of small-scale and can be found in Amazon ([www.amazon.com](http://www.amazon.com)), TripAdvisor or MovieLens ([movielens.umn.edu](http://movielens.umn.edu)), just to mention few. Larger scales can be easily shown with sliders, as done in Jester’s joke recommender ([eigentaste.berkeley.edu](http://eigentaste.berkeley.edu)). Having more values, people can express their opinion in a more detailed way, letting the recommender rank the items with more precision, but on the other hand the rating scale has some influence in the collected ratings [22]. Moreover, the different rating values could be interpreted differently by users, mapping their opinions to rating values in a different way one from the other: some people could be more “optimistic” and rate all good items with 5 stars, while others could be more “pessimistic” and reserve the 5 stars to items that are really awesome [55]. The action of mapping opinions to rating values require the user different cognitive load according to the scale used.

Past studies about rating scales in computer interfaces identified the 5-star rating scale as the best one according to users when rating movies or product reviews through a desktop device, immediately followed by thumbs-up/thumbs-down [68, 10, 22]. The 5-star rating scale has the advantage of letting people assign a neutral rating when they do not have a clear opinion about an item, without forcing a decision which may be misleading [20]. Also, the way in which the scale is represented (e.g., with only positive values or centered in 0 with negative values for bad ratings) has an impact on how people select the rating that represents best their opinion [3].

### 2.3.2 Collecting ratings

Given a rating model to apply, providing precise and accurate recommendations requires first and foremost data, that is, user ratings of items. These data can be used to both provide recommendations at system runtime or to train and fine-tune the recommendation algorithm at development time. Both scenarios are important in the context of recommender systems. For the former scenario, data can be collected directly from the users using the system, consuming recommendations, and providing their own ratings and feedback. For the latter scenario, data collection is typically more complex, in that the system to be developed (which would allow the users to express ratings) is not yet in place. It is therefore needed to gain access to useful data without involving the users of the future system.

One way of achieving this that has gained momentum over the last years is *crowdsourcing*, i.e., outsourcing of a piece of work (e.g., the rating of places) to an unknown group of people via an open call for contributions [30]. Typically, this is done via the Web through so-called crowdsourcing platforms like Amazon Mechanical Turk ([www.mturk.com/mturk](http://www.mturk.com/mturk)) or CrowdFlower ([www.crowdflower.com](http://www.crowdflower.com)).

Ratings are usually collected online, advertising a webpage where people can participate to a survey asking them to provide a set of ratings for places. In this respect, crowdsourcing has two key pros: i) the survey can be advertised to many *workers* around the world, and ii) respondents can be motivated offering rewards (typically small amounts of money).

The rewards are great because they let researchers collect a huge amount of data at relatively low cost and in short time. On the other hand, rewards motivate respondents to cheat, providing random or wrong answers, to complete the *task* as fast as possible, increasing their personal hourly income. Since data are needed for testing the researchers' work, they do

not want to pay for bad data, so quality controls are needed and workers providing bad data should not be paid.

We already mentioned that crowdsourcing platforms provide respondents from around the world. What if we want our survey to be answered only by people with specific skills? We need to filter workers and let them answer only if they can prove that they have the needed skills. These two problems of worker selection and cheating detection are explained in the following, together with the solutions suggested in the literature.

### **Worker selection**

In the literature, the problem of identifying the right workers for completing a task is usually solved in two ways: i) putting restrictions on the workers according to some information available in their profiles; ii) requiring to complete a qualification task before being able to access the actual task.

In crowdsourcing platforms, the requesters can hire people from all over the world to complete a task, when the task is in an internationally known language like English. This can be great for creative tasks, where a different culture could lead to unexpected good results, but in other cases the different culture and environment condition could make the task puzzling or even offensive. For this reason many crowdsourcing platforms allow the requesters to define some characteristics the workers should have to be able to perform the task and these characteristics are usually related to the country the worker lives in and the languages she knows. Such information is contained in the worker profile and the verification is executed by the platform when the worker shows interest in the HIT (“Human Intelligence Task” in the terminology of Mechanical Turk).

Profile information is usually not enough for understanding whether a worker has the correct skills to correctly perform a task. For example, some tasks could require some specific knowledge in Math or in another specific

subject. Workers' skills could be learned by the crowdsourcing platform by tracking workers' performance on the solved tasks and by identifying which skills are requested to correctly solve the specific tasks. A framework for such kind of crowdsourcing platform has been presented in [60], named SmartCrowd. The defined crowdsourcing system automatically evaluates the accuracy of completed tasks, learns the skills of the worker from her answers and assigns tasks to workers by matching their skills with the ones required to solve the tasks. Still, the ability to check users' skills depends on the tasks they already completed: if our task require some specific skills it could be possible that no previous task gave such type of skill information.

More detailed and task-specific conditions can be verified through a qualification task. Workers are asked to complete first the qualification task, that is actually a separated HIT, which contains questions or activities specifically designed to identify if the worker has the needed capacities to perform the actual task. Only the workers that give the correct answers or perform the task well enough are allowed to access the actual task. For example, in [2] workers are asked to judge if a document is relevant to a search query. Since documents are articles about different countries, workers need to know some geography and this skill is tested with few questions in a qualification test. Another example can be found in [14], where the level of knowledge of a specific city is verified before allowing the worker to access the task. In this case the pictures of some less known POIs are shown to the worker, which has to give the correct name of them to show her deep knowledge of the city.

### **Cheating detection**

There are many studies about the quality of crowdsourced work, also in relation with experiments that evaluate information retrieval and recommender algorithms. Before thinking on methods to identify if a worker is

cheating, it is important to design the task in a way that cheating is as discouraged as possible, possibly by making it more effortful than giving a truthful answer. In [36] the authors show how the number of workers trying to cheat is decreased by designing the task in a way that makes cheating at least as effortful as giving truthful answers, while Eickhoff and de Vries in [16] found that cheaters are discouraged by tasks that have the following characteristics: require creativity and abstract thinking; have many context changes; have small batch sizes. They also found that previous task acceptance, as was implemented at CrowdFlower and its available channels in 2010, is not enough to understand workers' reliability since it was easily cheated too.

In many cases, a good task design is not enough for discouraging cheating and the truthfulness of the collected information need to be checked using different mechanisms. The most common ones are the followings:

- *Gold answers*, i.e. extra questions with verifiable answers: if people are just giving random answers without reading the questions, they will likely give the wrong answers to these verifiable questions. If it is possible, the questions should refer to the content of the items being rated, so the correct answer also checks that the user is actually paying attention to what she is reading. In [29] the workers have, for example, to answer questions about how many stops they noticed in a video: stops were artificially added and there was only one correct answer known by task designers.
- *Verification and control questions*: These are used mainly when the answer of the user is very subjective. The correctness of the answer is checked by asking the same question twice and comparing the results. In [39] this result is obtained asking twice to classify the mood of some music. The worker cannot understand the music is the same she

already classified if she is not paying attention, so only trustful workers can give twice the same answer and pass the verification question. A variant is made of control questions: they are used to verify if the user understood the topic related to the task and if she is paying the right attention to what she is doing [17, 29]. Qualification tests are also usually deployed to verify that the workers are knowledgeable about the tasks they will be performing [2, 14].

- *Majority decision*: This technique allows one to verify if the answers by one worker are in line with the answers by other workers. This is true if there is only one correct answer, that we expect to be given by the majority of the workers [2].
- *Control group*: The idea here is to crowdsource the verification procedure, that is, to create another task in which workers will have to check whether the answers to the first task are plausible/correct. This can be used again only if there is a commonly known good (objective) answer [25].
- *Log analysis*: The interactions of the worker with the task interface are logged and checked in a second moment. If the worker is spending too much or too few time reading instructions or choosing/building the answer, she is considered a cheater. In [29] this technique is applied together with gold answers and content questions.

The quality of data collected through crowdsourcing can be influenced also by simple feedback given to the worker while she performs the task. In [66] the authors tested the effect of different motivational messages on the quality of workers answers. The authors found that better results can be obtained by telling the worker that her work will be verified and that if the answers are wrong and too different from the ones of the other workers,



she will be banned. Le et al. in [38], instead, find that better results are obtained by giving feedback to workers during their training task. In this way good workers learn how to complete the task in the best way, while cheaters abandon the task since their attacks are identified and make them spend more time.

### **Crowdsourcing subjective opinions**

In recent years, many work have been done on improving the data quality for crowdsourcing. Most such work focused on the joint inference of true labels of items and worker reliabilities after the data have been collected [13, 31, 33, 78, 25]. However, all these methods suffer from two drawbacks. First, they assume the presence of one ground truth, even if it is unknown, which is clearly not the case for subjective crowdsourcing tasks such as the ones we are concerned within this thesis (i.e., rating of items). Second, all these methods are generally post-processing methods, so they cannot be easily used during task assignment to improve data quality as more tasks are being performed.

To address the first issue, namely, the presence of a single ground truth, Tian and Zhu [69] studied the problem of worker reliability in crowdsourcing tasks in the case where more than one answer could be valid and reasonable. They directly modeled worker reliability and task clarity without the help of gold standards. Their model was built on two mild assumptions on the grouping behavior that happens in schools of thought: 1) reliable workers tend to agree with other workers in many tasks; and 2) the answers to a clear task tend to form tight clusters. Following this idea, they developed a low-rank computational model to explicitly relate the grouping behavior of schools of thought, characterized by group sizes, to worker reliability and task clarity.

There have been various attempts to integrate worker reliabilities or skills with task assignments in crowdsourcing platforms. Li and Zhao and Fuxman [42] proposed a crowdsourcing platform that can automatically discover, for a given task, if any group of workers based on their attributes have higher quality on average; and target such groups, if they exist, for future work on the same task. Satzger and Psai and Schall and Dustdar [62] proposed to use auctions to map tasks to workers in the crowd where a requester defines a maximum amount of money she is willing to pay. They then deployed an auctioning mechanism that provided a beneficial distribution of tasks to the available resources. Karger and Oh and Shah [35] considered a general model of crowdsourcing tasks and posed the problem of minimizing the total price (i.e., number of task assignments) that must be paid to achieve a target overall reliability. They gave a new algorithm for deciding which tasks to assign to which workers and for inferring correct answers from the workers' answers.

Ho and Vaughan [27] explored the problem of assigning heterogeneous tasks to workers with different, unknown skill sets in crowdsourcing markets such as Amazon Mechanical Turk. Inspired by research on the online adwords problem, they presented a two-phase exploration-exploitation assignment algorithm to allocate workers to tasks in a way that maximizes the total benefit that the requester obtains from the completed work. The same authors together with Vaughan [26] investigated the problem of task assignment and label inference for heterogeneous classification tasks. By applying online primal-dual techniques, they derived a provably near-optimal adaptive assignment algorithm and showed that adaptively assigning workers to tasks can lead to more accurate predictions at a lower cost when the available workers are diverse.

Roy, Lykourantzou, Thirumuruganathan, Amer-Yahia and Das [60] proposed in a vision paper to rethink crowdsourcing as an adaptive process

that relies on an interactive dialogue between the workers and the system in order to build and refine worker skills while tasks are being completed. In parallel, as workers complete more tasks, the system learns their skills more accurately, and this adaptive learning is then used to dynamically assign tasks to workers in the next iteration. This dialogue between the system and workers resembles the dialogue between users in trust-based systems [50]. In the same way in which in an e-commerce website buyers rely on the sellers' reputation and on the past interactions to understand how much they can trust them, in a crowdsourcing platform the task requesters use the learned skills of workers to understand how much they can trust them. However, the meaning of worker skill is different from trust, as it includes also the specific capabilities of a worker.

## 2.4 Conclusion

We have seen how research has been active in the different areas related to recommendations of places for leisure activities, but there are still questions that remain without an answer.

Although recommender systems have been studied for a while, their application in the context of leisure activities still needs more understanding and the development of new applications like Foursquare local search and The Fork is a clear indicator of the activity and evolution in this area. In research many personalized recommendation algorithms have been developed and studied, but their application is still limited, in particular in the context of leisure activities.

We have seen that according to literature SVD is one of the best recommendation algorithms, but these evaluations are based mainly on movie datasets, where there is no context to consider and people are less influenced by others. A deeper understanding of the application of these algo-

rithms for recommending places for leisure activities is needed. Moreover, these evaluations are mainly based on offline experiments (i.e. simulating the requests and evaluating the results with objective measures such as precision and mean average error), while user-based evaluations are fundamental to understand how much people are satisfied of the recommended items.

The identification of the best rating scale, that provides the correct amount of options to correctly collect all the shades of people tastes and at the same time is easy to use, is a problem that has been already tackled. For desktop devices, the 5-star rating scale has been identified as the best one, but at the best of our knowledge, no one explored the adoption of rating scales on mobile devices, where real estate and touch interface make the interaction different.

When volunteering is not providing the needed amount of ratings or when a new dataset is needed to answer some interesting research questions, crowdsourcing platforms could be the answer: they let you easily reach a crowd willing to collaborate at the cost of a reward (balanced with the time needed to complete the task). Crowdsourcing has been used to complete tasks easy for humans and hard for computers, such as labeling images, and different techniques for identifying the correct solutions between the many provided have been developed. The main difference between usual tasks and rating collection lays in the subjectivity of the information provided in this last type of task. The techniques for identifying cheaters in crowdsourcing platforms usually rely on the presence of a ground truth, i.e. an objective correct answer which all trustful workers agree on. In the case of rating collection this ground truth does not exist, and novel quality-control techniques are needed to respect the subjectivity of workers' opinions.

## Chapter 3

# The Role of Friends in Decision Making

### 3.1 Introduction

In Section 1.1 we have seen how people feedback about the items they know about can be collected in different ways. Between the many implicit and explicit user feedbacks we can use as source of people opinions, we decided to focus on ratings: people select the value that better represent their opinion from the provided scale. Assigning a rating to an item requires only few seconds, in which the user reminds her last experience with that item and maps her level of liking to a value within the scale presented to her. Being a form of explicit feedback, it has the advantage that the system is collecting the real opinion of the user and does not need to interpret user's behaviour to extrapolate her tastes, as it happens when collecting implicit feedback. We decided to not consider reviews and other more complex explicit feedbacks because of their effort: the more time is required for a single feedback, the fewer feedbacks the user is willing to leave. Another advantage of collecting ratings is that their values can be immediately consumed by recommender algorithms, without the need for special conversions.

Friends and relatives are important in the leisure environment as we usually spend leisure time with them. In the last years we have seen an increasing usage of social networks. People like to meet friends online and share with them the interesting facts of their life. Thanks to social networks, now relationships between friends are easily accessible without the need to ask directly people about them.

In this chapter we study whether the relationships between users as stated in social networks can be used to improve recommendation quality. The question behind this study is the following: since in real life friends' recommendations about what to do in leisure time are very important, do friends' tastes better represent the requester's ones? Collaborative filtering algorithms use the opinions of users with tastes similar to the requester to predict how much the requester would like an unknown/un-rated item. What happens if instead of considering "similarity" relationship we consider "friendship" relationship as it is stated in social networks?

For the following evaluation, we focused our attention on restaurants. Going out for dinner is a very common activity and almost everyone knows at least some of the restaurants available in her city. As most of the other leisure activities, it is usually performed with some companions, i.e. friends, the partner, the family or other relatives. The choice between restaurants is hard as their characteristics, such as cuisine type, location and price, are usually not enough to understand the type and quality of the experience we can have there. As for all the other places for leisure activities, the opinions of others are important to understand the quality of the service offered and be able to identify the right place for us.

## 3.2 Sample Population and Comparisons with Baselines

To study the effectiveness of ratings, we collected a set of ratings about restaurants in Trento, Italy (694 ratings from 90 users, over a total of 75 restaurants we considered in the study, taken from TripAdvisor’s top 75 ranked restaurants). Ratings were on 1-5 scale as TripAdvisor does. However, our population was people who live in Trento and therefore know the restaurant scenes rather well. Furthermore, each person rated on average just short of 8 places in the same town. We do not have in our survey ratings from occasional visitors who has only been in one or two places, and we do not have ratings from restaurant owners or their competitors. The population in the study is composed of relatively young adults that have a Facebook account (we required a Facebook login to perform analysis of the effectiveness of friend-based recommendations). This means that our population was mainly composed of young adults, but we don’t consider this to be a bias because the sample is also the same as the population we target.

We take therefore the results of the ranking as a representation of the average opinion of locals.

As a first part of the analysis, we studied how well results from popular travel guides match our ground truth. We took as notable example the Lonely Planet 2012 guide for Italy [24]. The result is shown in Table 3.1. The guide lists five restaurants. One of them is not among TripAdvisor’s top 75 and therefore is not in our test list. The other four are, interestingly, in very different position in the three rankings. Lonely Planet considers them to be the four most interesting restaurants to visit, while for our ranking they are more or less evenly distributed among the top half but are quite far from the top 4 (with one exception), and the results are analogous

Table 3.1: LonelyPlanet recommended restaurants for Trento

<b>Lonely Planet</b>	<b>Our ranking (out of 75)</b>	<b>TripAdvisor ranking</b>
Scigno	5	10
Uva e Menta	15	55
Tre Garofani	20	7
Pedavena	37	47

Table 3.2: TripAdvisor top 10 recommended restaurants for Trento

<b>Restaurant</b>	<b>TripAdvisor</b>	<b>Our ranking</b>
Due Spade	1	1
Duo	2	35
Vecchia Macina	3	35
Albert	4	35
Margon	5	63
Madruzzo	6	5
Tre Garofani	7	20
Gusto	8	54
Pineta	9	69
Scigno	10	5

for TripAdvisor (TripAdvisor and our rankings are also very different, but we'll get back to this later). We can say that Lonely Planet did not, in this case, guess the preferences of our test users population.

As another baseline analysis, we consider TripAdvisor. Instead of using correlation coefficients we take a very pictorial way of performing the analysis: we consider the effectiveness of the top 10 rated restaurants on TripAdvisor (assuming we would look at those to select where to go) and see how well we would do with that.

As Table 3.2. shows, in three cases we would pick one of the our top 10 restaurants, while in the other cases we pick places that are rather far down the ranking. Indeed, given that we have a population of 75 restaurants, 6 out of these top 10 are around or below that midpoint!



We now turn to look at how we can best use such ratings to perform accurate and personalized recommendations. Specifically, we first see how our overall ratings are accurate in giving recommendation to a user, and then compare this with recommendations given by considering only the requester friends' rating, by considering ratings from people with similar taste as the requester (that rated similarly), and last by considering the ratings by similar friends.

### 3.3 Formal Experiment Definition

Let  $\mathcal{U}$  be the set of all users and  $\mathcal{P}$  the set of all the places. Liked represents the relation  $(user, place) \in \mathcal{U} \times \mathcal{P}$  of places users rated positively, and Disliked the relation  $(user, place) \in \mathcal{U} \times \mathcal{P} \setminus \text{Liked}$  of places users rated negatively. FriendOf denotes the relation  $(user, user) \in \mathcal{P}^2$ . Given these basic elements we define:

$$\text{Known}(u) = \{p \in \mathcal{P} \mid \text{Liked}(u, p) \vee \text{Disliked}(u, p)\},$$

$$\text{Unknown}(u) = \mathcal{P} \setminus \text{Known}(u)$$

to denote the sets of places the user have rated and the ones yet to discover.

For the scoring function, we compute the average rating of the network in consideration:

$$\text{score}(p, \text{Net}(u)) = \frac{\|\text{Likes}(p, \text{Net}(u)) - \text{Dislikes}(p, \text{Net}(u))\|}{\|\text{Net}(u)\|}$$

$$\text{Net}(u) = \{u' \in \mathcal{U} \mid \text{sim}(u, u') > \delta\}.$$

$$\text{Likes}(p, \text{Net}(u)) = \bigcup \{u' \in \text{Net}(u) \mid \text{Liked}(u', p)\}$$

$$\text{Dislikes}(p, \text{Net}(u)) = \bigcup \{u' \in \text{Net}(u) \mid \text{Disliked}(u', p)\}$$

Finally, we define the recommendation of places  $p$  to a user  $u$  as:

1.  $\text{Rec}(u, k) \subseteq \text{Unknown}(u)$ ,
2.  $|\text{Rec}(u, k)| = k$ ,
3.  $\forall p \in \text{Rec}(u, k) \forall p' \in (\text{Unknown}(u) \setminus \text{Rec}(u, k))$   
 $(\text{score}(p) \geq \text{score}(p'))$ .

### 3.4 Definition of Recommendation Strategies

We tested four different settings of the above definition to compare the effectiveness of the different networks to recommend places:

#### Recommendation based on overall popularity

In this setting we consider the ratings from all users to compute the score ( $\text{score}_o$ ):

$$\text{sim}_o(u, u') = 1, \forall u, u \neq u'$$

#### Recommendation based on friends

This setting makes use of the network of friends to compute the score ( $\text{score}_f$ ):

$$\text{sim}_f(u, u') = 1, \exists \text{FriendOf}(u, u')$$

#### Recommendation based on similar users

This setting makes use of people with similar taste to compute the score ( $\text{score}_{rt}$ ). The similarity function in this case is defined as:

$$\text{sim}_{rt}(u, u') = \frac{\|\text{Co\_liked}(u, u') \cup \text{Co\_disliked}(u, u')\|}{\|\text{Known}(u) \cap \text{Known}(u')\|}$$

$$\text{Co\_liked}(u, u') = \bigcup \{p \in \mathcal{P} \mid \text{Liked}(u, p) \wedge \text{Liked}(u', p)\}$$

$$\text{Co\_disliked}(u, u') = \bigcup \{p \in \mathcal{P} \mid \text{Disliked}(u, p) \wedge \text{Disliked}(u', p)\}$$

### 3.5. EVALUATION OF THE DIFFERENT RECOMMENDATION ALGORITHMS

Table 3.3: Evaluation of the different scoring functions,  $k = 5$

Score function ( $k = 5$ )	Precision		Recall	
	Av.	sd	Av.	sd
$score_o$	0.09	0.15	0.08	0.12
$score_f$	0.16	0.21	0.12	0.17
$score_{rt}$	0.12	0.18	0.08	0.14
$score_{sf}$	0.25	0.26	0.14	0.16

Table 3.4: Evaluation of the different scoring functions,  $k = 10$

Score function ( $k = 10$ )	Precision		Recall	
	Av.	sd	Av.	sd
$score_o$	0.11	0.12	0.16	0.16
$score_f$	0.17	0.16	0.22	0.20
$score_{rt}$	0.13	0.13	0.17	0.21
$score_{sf}$	0.22	0.21	0.25	0.28

#### Recommendation based on similar friends

This setting makes use of friends with similar taste to compute the score ( $score_{sf}$ ). The similarity function in this case is defined as:

$$sim_{sf}(u, u') = sim_{rt}(u, u'), \exists \text{FriendOf}(u, u')$$

### 3.5 Evaluation of the Different Recommendation Algorithms

We evaluated each recommendation strategy by computing all variations of the *score* function for all users  $u$  on every place  $p \in \mathcal{P}$ , dropping the place from the set of known places  $Known(u)$  as the initial condition. Then we took the top  $k$  places and compared this list with how many places the user actually liked computing *precision* and *recall*. The summary of the results can be seen in Table 3.3 and Table 3.4.

In the evaluation we use the following definition of *precision* and *recall*:

$$precision = \frac{\|\mathbf{Tp}(u)\|}{\|\mathbf{Tp}(u)\| + \|\mathbf{Fp}(u)\|}$$

$$recall = \frac{\|\mathbf{Tp}(u)\|}{\|\mathbf{Fp}(u)\| + \|\mathbf{Fn}(u)\|}$$

$$\mathbf{Tp}(u) = \bigcup\{p \in \text{Rec}(u) \mid \text{Liked}(u, p)\}$$

$$\mathbf{Fp}(u) = \bigcup\{p \in \text{Unknown}(u) \mid \text{Disliked}(u, p)\}$$

$$\mathbf{Fn}(u) = \bigcup\{p \in \text{Known}(u) \setminus \text{Rec}(u) \mid \text{Liked}(u, p)\}$$

The results suggest, not surprisingly, that recommendations coming from similar users (based on rating behaviour) performs better than overall popularity. More interestingly, recommendations coming from friends, and specially if reduced to similar friends, outperforms considerably the other algorithms in both precision and recall. These results are promising and points to the potential of balancing personal tastes (in this case captured by rating behaviour) with the real social context (friends).

We should interpret the higher precision as higher chances of finding interesting restaurants, and the higher recall as the reduced feeling of missing out something.

### 3.6 Conclusion

From this analysis we learn that there is space for improvement in recommending good places to eat (and, likely, the same applies to other categories of places for leisure activities), as we have seen that travel guides and web sites are not satisfying the locals involved in our experiment. Specifically, an interesting take home point is that not only unbiased ratings from locals differ from those of travel guides, but also that considering friendship and

similarity have a profound effect on the accuracy of the recommendations. We assume that the friendship effect is not so much related to being friends but rather to being in the same age group, but we did not have the chance to verify this. Recommendations based on similar friends provide by far the best result (almost three times better than overall ratings), combining (we believe) age with taste similarity.



# Chapter 4

## How Purpose Influences Opinion

### 4.1 Introduction

In this chapter we study how people decide what activity to perform during leisure time, and in particular in which places they can perform the chosen activity, focusing our attention to typical activities that are performed on a standard evening in three different cities in the world: drinking aperitif in a bar and having dinner at a restaurant in Trento, Italy; having dinner at a restaurant and drinking some beer in a pub in Asunción, Paraguay; having dinner at a restaurant and dancing in a club in Tomsk, Russia. We discovered that places for performing these activities are chosen differently according to the kind of companions people are spending their leisure time with, and in which situations price/quality ratio is important.

Recommendations are analysed too, finding that recommendations can be computed also across different activities: knowing user's tastes about restaurants, we can recommend which places she can like for the other activity considered in her city. This discovery leads to the possibility of profiling users based on their lifestyle, allowing us to extend the recommendation service to all different kinds of leisure activities maintaining the high quality.

The main contributions of this chapter can be summarised as:

- a large scale study on three different cities around the world, gathering a total of 9820 ratings from 162 local people.
- an analysis of factors affecting people’s decision and preferences in participating in leisure activities, considering the *social context*, *preferences across activities* and the *effect of individual’s purpose* in engaging in an activity.

We have seen in Chapter 3 how for Trento’s people Facebook friends are a useful user-base for collaborative filtering-based recommendations of restaurants. In this chapter we extend the previous experiment in three directions: i) we run it in 3 different cities around the world (Trento in Italy, Asunción in Paraguay, and Tomsk in Russia); ii) we considered two different leisure activities in each city, one of which is again restaurants while the other is different for each city; iii) for each place, users were able to specify 4 different ratings according to different purposes that can be accomplished with an activity.

## 4.2 Formal Definitions

Before describing the study in details, we start by providing some formal definitions. The formulae we are presenting here extends the ones already explained in Section 3.3. In this case, we are not considering generic places, but specific businesses (restaurants, bars, pubs and clubs), so we will use these two words as synonyms.

Let  $\mathcal{U}$  be the set of all users and  $\mathcal{B}$  the set of all the businesses where activities can be performed. Let  $\mathcal{A}$  be the set of all activities, and  $\mathcal{P}$  the set of purposes that can be accomplished with an activity. **Liked** represents the relation  $(u, b, a, p) \in \mathcal{U} \times \mathcal{B} \times \mathcal{A} \times \mathcal{P}$  of businesses users rated positively for a given activity on a specific purpose, and **Disliked** the relation  $(u, b, a, p) \in$



$\mathcal{U} \times \mathcal{B} \times \mathcal{A} \times \mathcal{P} \setminus \text{Liked}$  of businesses users rated negatively. As an example, this definition can capture a situation in which a user ( $u$ ) likes a given restaurant ( $b$ ) to have dinner ( $a$ ) with her partner ( $p$ ).

Given these basic elements we define:

$$\begin{aligned} \text{Rated}(u, b, a, p) &= \text{Liked}(u, b, a, p) \vee \text{Disliked}(u, b, a, p) \\ \text{Known}(u, a) &= \{b \in \mathcal{B} \mid \exists p \in \mathcal{P}, \text{Rated}(u, b, a, p)\} \\ \text{Unknown}(u, a) &= \mathcal{B} \setminus \text{Known}(u, a) \end{aligned}$$

In the previous chapter we have seen how friends are better predictors of users tastes, while this time we focus only on similarity. Similarity is defined as a measure of how much users' tastes are similar and is computed counting how many times the two users gave a similar rating to the same places:

$$\text{sim}(u, u', a) = \frac{\|\text{Corated}(u, u', a)\|}{\|\text{Known}(u, a) \cap \text{Known}(u', a)\|}$$

$$\begin{aligned} \text{Corated}(u, u', a) &= \bigcup \{b \in \mathcal{B} \mid \exists p \in \mathcal{P}, \\ &\quad \text{Liked}(u, b, a, p) \wedge \text{Liked}(u', b, a, p) \vee \\ &\quad \text{Disliked}(u, b, a, p) \wedge \text{Disliked}(u', b, a, p)\} \end{aligned}$$

For the scoring function, we compute the average rating of the network  $\text{Net}(u) \subseteq \mathcal{U}$  in consideration:

$$\text{score}(u, b, a, p) = \frac{\|\text{Likes}(u, b, a, p)\| - \|\text{Dislikes}(u, b, a, p)\|}{\|\text{Net}(u)\|}$$

$$\text{Net}(u) = \{u' \in \mathcal{U} \mid \exists a' \in \mathcal{A}, \text{sim}(u, u', a') > \delta\},$$

$$\text{Likes}(u, b, a, p) = \bigcup \{u' \in \text{Net}(u) \mid \text{Liked}(u', b, a, p)\}$$

$$\text{Dislikes}(u, b, a, p) = \bigcup \{u' \in \text{Net}(u) \mid \text{Disliked}(u', b, a, p)\}$$

On this foundation, we study how different factors influence user preferences by analysing how effective they are in recommending places users

would probably like. To this end, we define the recommendation of businesses  $b$  to a user  $u$  to perform an activity  $a$  with a purpose  $p$  as:

1.  $\text{Rec}(u, a, p, k) \subseteq \text{Unknown}(u, a)$ ,
2.  $|\text{Rec}(u, a, p, k)| = k$ ,
3.  $\forall b \in \text{Rec}(u, a, p, k)$   
 $\forall b' \in (\text{Unknown}(u, a) \setminus \text{Rec}(u, a, p, k))$ ,  
 $(\text{score}(\text{Net}(u), b, a, p)) \geq \text{score}(\text{Net}(u), b', a, p)$ .

### 4.3 Extending the Experiment of Chapter 3

The study we present in this chapter extends the previous one in three directions: i) the geographic area is extended to three different cities around the world; ii) in each city we asked ratings not only for restaurants but also for another activity that is usually done before or after going out for dinner; iii) for each place people were able to specify four different marks according to different purposes. The experiment was run in Trento (Italy) with bars for aperitif as second type of places, Asunción (Paraguay) with pubs, and Tomsk (Russia) with clubs. In all cases, we considered the same four purposes: one mark was dedicated to the price/quality ratio and the other three was related to the different types of companions people can spend their leisure time with, which are tourists, friends and their partner.

A website guided users through the procedure for rating the different places available for their city, requesting to access using their Facebook account (from which friends relationships are collected) and requesting at least five marked places in the first activity in order to access the list prepared for the second one. The ratings were collected using the thumbs-up/thumbs-down scale plus the neutral option. The website was published the 16<sup>th</sup> of November 2012 and collected people's preferences for two

months. It was advertised mainly through social networks, but in Trento some posters were also hung up in the university’s buildings. For Trento we asked to mark places for aperitif first, choosing between the 30 provided, and restaurants after, with a list of 67 places. 49 people participated to the study, leaving 2700 marks. For Asunción we started with a list of 254 restaurants, followed by 43 pubs and bars, collecting a total of 6100 marks from 97 people. We started from restaurants for Tomsk too, with a list of 32 places, followed by 12 clubs. We reached 16 people that left 1020 marks. The website was designed in a way that it was very easy to filter all the available places and find the ones that are known.

## 4.4 Understanding Preferences Across Activities

In this study, we focused on understanding whether it was possible to extend recommendations across different activities: knowing a user’s taste for restaurants, is it possible to recommend her another activity, and a place where to perform it, using only her taste for restaurants?

Going after that question, we collected ratings for two typical activities ( $a_1, a_2$ ) in our three target locations. Then, for each user ( $u$ ), we created a dataset without all her ratings for the second activity, leaving  $\text{Known}(u, a_2) = \emptyset$ . On this dataset, we calculated the recommendations assuming the following definition of network:

$$\text{Net}(u) = \{u' \in \mathcal{U} \mid \exists a' \in \mathcal{A}, \text{sim}(u, u', a') > \delta\},$$

which builds a set of users sharing similar tastes in  $a_1$ , since all the user ratings for  $a_2$  were removed. In doing so, we are recommending places for  $a_2$  (e.g., going clubbing) using the network of users with similar taste for  $a_1$  (e.g., going out for dinner). Borrowing the previous definition of similarity, we say that user  $u$  is similar to user  $u'$  if they agree in at least 70% of their ratings ( $\delta \geq 0.7$ ).

Table 4.1: Precision of the recommendations

	$Precision(10)$	$Precision(\ \mathcal{P}\ )$
Trento	0.85	0.81
Asunción	0.922	0.893
Tomsk	0.82	0.84

In order to understand how good these recommendations are, we computed precision for the top-10 places and the full list. Given  $\mathcal{R}(u, b, p, n)$  as the set of recommendations computed for user  $u$  on business  $b$  for purpose  $p$ , where  $n$  is the position in the ranking, and  $\mathcal{M}(u, b, p)$  as the set of marks given by user  $u$  on business  $b$  for purpose  $p$ , the precision is defined as

$$Precision(u, p, n) = \frac{\|Good(u, p, n)\|}{\|Good(u, p, n)\| + \|Bad(u, p, n)\|}$$

where

$$Good(u, p, n) = \{b \in \mathcal{B} \mid R(u, b, k) \equiv M(u, b) \wedge k \leq n\}$$

$$Bad(u, p, n) = \{b \in \mathcal{B} \mid R(u, b, k) \neq M(u, b) \wedge k \leq n\}$$

Recommendations were computed for all 4 different purposes, only for the users that had a similar group containing at least 3 users ( $\mathbf{Net}(u) \geq 3$ ). The resulting precision for the top 10 recommendations and of all of them, averaged by city, are reported in table 4.1.

As can be seen, the precision of such recommendations is high. For the places that the considered user already rated, the recommendations usually matched the user's rating, with very few errors that appeared mainly for users with a too wide similarity group. The results suggest that similarities in the preferences of people for an activity can be extended to other activities, which points to potential of profiling users based on lifestyle.

Table 4.2: *Kendall*  $\tau$  distances for Trento’s restaurants

	T	F	P	Q
T	1.0			
F	0.146	1.0		
P	<b>0.652</b>	0.162	1.0	
Q	-0.106	<b>0.288</b>	-0.027	1.0

Table 4.3: *Kendall*  $\tau$  distances for Trento’s bars for aperitif

	T	F	P	Q
T	1.0			
F	0.287	1.0		
P	<b>0.696</b>	0.232	1.0	
Q	-0.007	0.227	0.011	1.0

## 4.5 Effect of Purpose on User Preferences

Given all the collected ratings, we first analysed the differences between the rankings resulting from the marks for different purposes. All places, divided by city and by activity, were ordered, for each purpose, according to the average mark and the number of marks received, building in this way four rankings for each activity in each city. We considered only the places that received at least 5 marks for each purpose, maintaining only the places that collected a minimum amount of the crowd’s opinion. The remaining places are 23 restaurants and 30 bars for Trento, 87 restaurants and 26 pubs for Asunción, 21 restaurants and 12 clubs in Tomsk.

*Kendall*  $\tau$  distance was used to compute the difference between the four rankings for each activity, counting the number of couples in the rankings that appear in the same order. The returned value is between 1 and -1, with 1 meaning that the two rankings are equal, while -1 means that they are completely the opposite. Naming  $C$  the number of concordant couples and  $D$  the number of the discordant ones, the metric is defined with the

Table 4.4: *Kendall*  $\tau$  distances for Asunción's restaurants

	T	F	P	Q
T	1.0			
F	0.383	1.0		
P	<b>0.502</b>	0.388	1.0	
Q	0.324	0.328	0.262	1.0

Table 4.5: *Kendall*  $\tau$  distances for Asunción's pubs and bars

	T	F	P	Q
T	1.0			
F	0.28	1.0		
P	<b>0.36</b>	0.194	1.0	
Q	<b>0.335</b>	0.169	0.261	1.0

following formula:

$$Kendall \tau = \frac{C - D}{C + D}$$

The *Kendall*  $\tau$  distance is reported in tables 4.2, 4.3, 4.4, 4.5, 4.6 and 4.7, where the purposes are summarised in the following way: T = Bringing tourists, F = Bringing friends, P = bringing the partner and Q = price/quality ratio.

As can be seen, in both Trento's tables 4.2 and 4.3 the highest similarity is between the rankings for *Bringing tourists* and *Bringing the partner*, while they are both different from *Bringing friends* and in particular from *Price/quality ratio*, that has almost half of the couples in the wrong order. The distance between *Price/quality ratio* and *Bringing friends* is slightly less, giving a hint that going out with friends the quality and price of food and beverages are taken into consideration more than when spending time with tourists and the partner.

In Asunción, tables 4.4 and 4.5, there is still a higher similarity between *Bringing tourists* and *Bringing the partner*, but their distance to *Bringing friends* and *Price/quality ratio* is much less. Moreover, both *Bringing*

---

#### 4.5. EFFECT OF PURPOSE ON USER PREFERENCES

---

Table 4.6: *Kendall*  $\tau$  distances for Tomsk’s restaurants

	T	F	P	Q
T	1.0			
F	0.019	1.0		
P	<b>0.476</b>	-0.067	1.0	
Q	-0.133	<b>0.429</b>	-0.029	1.0

Table 4.7: *Kendall*  $\tau$  distances for Tomsk’s clubs

	T	F	P	Q
T	1.0			
F	0.182	1.0		
P	<b>0.454</b>	<b>0.545</b>	1.0	
Q	0.0	0.273	0.242	1.0

*friends* and *Price/quality ratio* are almost equally distant from all the other rankings, showing that there is no stronger correlation between them. Moving to pubs and bars, we can see that the similarity between *Bringing tourists* and *Bringing the partner* is not present here, with only *Bringing friends* slightly more distant from the other rankings.

In Tomsk, tables 4.6 and 4.7, the rankings are again divided in two groups, with *Bringing tourists* and *Bringing the partner* on one side and *Bringing friends* and *Price/quality ratio* on the other. Looking at the *Kendall*  $\tau$  distance for club rankings, the situation is different. Here the closest rankings are *Bringing friends* and *Bringing the partner*, immediately followed by the already known couple *Bringing tourists* and *Bringing the partner*. Despite this, *Bringing friends* and *Bringing tourists* are distant.

Summarizing all the results, we have seen that people preferences are sensitive to the companion (e.g., partner, friends, tourists) for which they look for different features. In particular, most of the times going out with friends results in different choices than going out with the partner

Table 4.8: Comparison of purpose-based ranks with TripAdvisor on Trento’s restaurants

	TripAdvisor	T	F	P	Q
Le due spade	1	13	18	12	23
Duo tapas bar	2	3	14	3	15
Loto	3	1	4	1	17
Niky’s	4	4	20	7	18
Oro stube	5	7	13	9	9
Welcome India	6	14	6	4	13
Rosa d’oro	7	8	1	6	7
Il cappello	8	6	21	5	16
Trattoria Piedicastello	9	9	8	19	8
Uva e menta	10	2	5	2	3
Da Andrea	-	16	7	15	1

or tourists, and sometimes goes together with higher attention to the price/quality ratio. As we have seen there are some exceptions to this generalization, specially for Asunción’s pubs where the price/quality ratio is considered more when choosing places where to bring tourists. These differences depending on the location can be intuitively be related to the cultural and economical aspects.

### Comparison with an existing service

One of the most popular services for recommendations of restaurants is TripAdvisor. We have locals’ knowledge of Trento’s restaurants according to different purposes: how does TripAdvisor’s rank compare with them? We selected the 23 restaurants in Trento that received at least 5 ratings for each purpose and ranked them according to their average rating. We obtained in this way 4 different rankings, and we compared them with the rank of the same restaurants computed by TripAdvisor: this means that we got the TripAdvisor full rank of all Trento’s restaurants and we removed the restaurants not present in our list.



Table 4.9: *Kendall*  $\tau$  distances with TripAdvisor for Trento’s restaurants

	T	F	P	Q
TripAdvisor	0.431	0.020	0.478	-0.186

Table 4.8 shows the position of the top-10 restaurants according to TripAdvisor in our 4 purpose-based ranks. The top-10 restaurants are in the top-15 positions for our *Bringing tourists* rank and in the top-19 positions for our *Bringing the partner* rank. Differently, the rank for *Bringing friends* is quite different and only half of the restaurants are still in the top-10 positions. The difference is even higher if we look at the fourth rank: the one dedicated to *Price/quality ratio*. In this case, the restaurant that is at the first place according to TripAdvisor is instead at the last one in our rank (23<sup>rd</sup> in a list of 23 restaurants) and the restaurant that is the best for *Price/quality ratio* is not even in the rank of TripAdvisor.

*Kendall*  $\tau$  distance confirms these first impressions (see Table 4.9). *Bringing tourists* and *Bringing the partner* have very high values, above 0.4, indicating that the differences are small, while the *Kendall*  $\tau$  distance of *Bringing friends* is very close to 0. The negative value for *Price/quality ratio* confirms our intuition of an opposite tendency of this rank with respect to TripAdvisor’s one.

Since TripAdvisor is a service dedicated mainly to tourists, these results are not surprising. It is providing a very good service to its main user base, and also locals confirm that the recommended restaurants are good for tourists. On the other hand, locals would not be satisfied by TripAdvisor’s recommendations if they want to use it when going out with friends, for example. This means that there is space for another restaurant recommender service dedicated to locals and better tailored to this goal.

According to our participants, many of the restaurants recommended by TripAdvisor do not provide a good price/quality ratio. By considering this

aspect, TripAdvisor could improve its service and make even tourists with low budget happier of the recommendations. We did not explore whether Tripadvisor's filter by price could improve user satisfaction for this specific purpose.

## 4.6 Conclusion

In this chapter we studied various aspects around user preferences that could be used to help users find activities and places they would actually like. We found that the knowledge about the users' preferences and neighbours on one specific activity (i.e., restaurants) can be used to recommend another related activity (i.e., a common saturday evening activity) even without any information about a specific user preferences on the second activity, reducing the cold start problem of collaborative filtering. Moreover, thanks to this we can provide better recommendations, extending the knowledge-base to different activities.

Moreover, we identified four different purposes that make people select places for leisure activities differently. We discovered that companions have a high influence, as the places identified as good for going out with tourists or the partner are different from the ones for going out with friends. Even price/quality ratio has an effect on the choice and a good ratio is considered more when going out with friends, maybe as people try to not spend too much this occasion.

Finally, we have seen the effect of purpose also with respect to TripAdvisor's rank of Trento's restaurants. As we expected, TripAdvisor's rank is closer to our *Bringing tourists* purpose: in fact, the service is focused on tourists and it does a good work for its goal. These results show the possibility to extend TripAdvisor by including also locals in the user-base and providing locals-focused recommendations.

# Chapter 5

## Purpose-orientation and Focus on Locals

### 5.1 Introduction

At the core of each recommender system there are two ingredients: first, the recommender algorithms that select candidate items; second, the data that provide the base for the recommendations [1]. In general, the better the algorithms and the more the data available, the better the recommendations. We have already seen in Section 2.2 how algorithms are the traditional focus of research.

Data is often represented by ratings on items, and literature on recommender algorithms typically uses standard datasets for the assessment of algorithms, such as the 5-star ratings for movies by *MovieLens* [7] or *Netflix* [7, 40]. Other common rating scales are unary (like), binary (thumbs-up/thumbs-down) and 3-value scales (thumbs-up/neutral/thumbs-down). Ratings come as values with no extra information about the experience or context of the user that led to the judgments, leaving the interpretation of the ratings to the recommender algorithms. In Chapter 4 we started studying the recommendation of leisure activities (drinking, dining, dancing), and we noticed that the quality of recommendations on items (in

our cases, restaurants, bars and clubs) strongly depends on the specific purpose of the activity: one place may be good for dining but not for drinking; another one may be good for a romantic dinner but not for one with friends. This kind of nuances is usually not captured by state-of-the-art recommender systems and, hence, cannot be used to better tune recommendations to users.

In addition, in the specific context of restaurants and dining, tourists have generally very limited knowledge of a new city and, therefore, most of the times rely on recommendations from family members or friends; if these are not available, tourists also like to rely on recommendations from *locals* [59]. They see locals as knowledgeable and trustworthy, since they know that locals know most of the available options, have been there themselves, and can provide personal recommendations for free. Again, this kind of knowledge from locals is typically lost in online tourist portals, which are mostly oriented toward and, hence, visited by tourists themselves. Good recommendation services specifically tailored to locals are still underrepresented.

These considerations raise the question whether data collected with i) special attention to locals and ii) information on specific usage purposes in mind can recommend restaurants with higher *precision* (that is, higher probability of recommending a place the user will actually like) than generic recommender systems, such as *TripAdvisor* for restaurants. In order to answer this question, this chapter studies the case of collaborative filtering algorithms and tests the following two hypotheses:

*H1: Data collected from locals and state-of-the-art, personalized recommendation algorithms produce recommendations of higher precision than generic recommender systems, such as TripAdvisor.*

*H2: Recommendations computed from purpose-specific data outperform TripAdvisor.*

The findings show that indeed taking these aspects into consideration can lead to improved recommendations with respect to the mainstream recommender in this domain. The findings also unveil insights that are particularly important to mobile recommender systems characterized by limited screen real estate.

## 5.2 Background

In a recommender system, users express their opinions about items in form of ratings. *Items* can be anything users can experience and can have an opinion about, while a *user* is any person that has experienced some of the items the system is focused on. Here we focus specifically on restaurants. These are physical establishments, so only people able to visit them can also experience them. From this perspective, a restaurant can have two kinds of customers: locals and tourists. *Locals* are people that live in the area, are familiar with the local cuisine and the restaurants, and can experience them several times. *Tourists* are visitors for business or leisure that generally have fewer chances to sample restaurants in a given area and are less familiar with the local cuisine.

When rating items, users assign only one rating per item, evaluating it according to their overall experience. *Multi-criteria* ratings, instead, ask users to add one rating for each of a set of predefined characteristics of the item. For example, in the case of restaurants, the criteria could be food quality, drink quality, service and popularity. This requires a user to consider the different aspects of her experience and give more ratings.

Orthogonally to this, a restaurant may be perceived differently depending on the *purpose* of the visit: the choice of a restaurant for a dinner with friends may differ from what we would choose for a romantic dinner or for a quick lunch. We already verified this hypothesis in our earlier research

reported in Chapter 4, where we also identified four main purposes: dinner with tourists, romantic dinner with the partner, dinner with friends and price/quality ratio (e.g., important for a lunch break).

## 5.3 Method

We study whether recommendations based on *purpose-specific data* collected from *locals* outperform recommendations computed from the typical data collected from tourists by tourist portals (TripAdvisor).

### 5.3.1 Data collection

We collected ratings using a 3-value thumbs-up/thumbs-down scale for each of the purposes identified in Chapter 4 (dinner with tourists, dinner with partner, dinner with friends and lunch break): users can specify whether they like or don't like a restaurant, or are neutral about it. In general, the thumbs-up/thumbs-down leaves less space for controversy than using 5 stars: the user just has to think about whether the item is good or bad, without having to think about how good (or how bad). The neutral rating prevents forcing the user to like or dislike an item if it is considered borderline.

As concrete dataset, in May 2014 we collected ratings for 50 restaurants in Trento, Italy. We selected this list considering the most popular restaurants according to TripAdvisor that are located in the city center and easily reachable by everyone. We enrolled participants by distributing fliers to locals, sending emails to friends and colleagues, and also involving a small group of university students. The participants (114) were asked to rate restaurants for each of the purposes (the 4 identified above) at a time. The process produced a total of 4706 ratings, with 1529 ratings for “dinner with tourists”, 1113 ratings for “dinner with the partner”, 1112 ratings

---

for “dinner with friends” and 952 ratings for “price/quality ratio”. The restaurants received a minimum of 4 ratings and a maximum of 112 ratings per purpose, while users added a minimum of 0 ratings and a maximum of 49 ratings per purpose, with an average of 11 ratings per purpose.

### 5.3.2 Recommendation algorithms

Computing purpose-specific recommendations poses challenges to the recommendation algorithms, as the algorithm has to work with multiple ratings per item per purpose. A first way to approach this multiplicity is to filter ratings to create one dataset for each purpose; in this way, only the information about the purpose the requester is interested in is used to compute recommendations. Another way is to merge all ratings from the different purposes and to compute aggregated ratings valid for all purposes, similarly to how multi-criteria ratings are handled by recommendation algorithms. A third solution is to learn user tastes using all collected data for all purposes and to compute ratings for each purpose individually; in this way, the whole information is used to extract taste features and to compute similarities between users or items, but only the ratings specific to a purpose for a user in a given instant of time are used for the prediction of ratings for unknown restaurants.

We followed this last solution. To handle the presence of 4 ratings per user-restaurant pair (one for each purpose), we split each user’s ratings for a restaurant into 4 purpose-specific restaurant-purpose pair, resulting in 200 ( $50 * 4$ ) items. In this way, all ratings can be considered in the computation of the model used by the algorithm (like building clusters for cluster-based collaborative filtering or computing matrix factorization for SVD), while only the restaurant-purpose pairs for the requested purpose are considered to build the rank when computing recommendations. To

adapt the algorithms to this behavior, we only need to extend them with a final filter of items by purpose.

For computing recommendations we selected four state-of-the-art, personalized, collaborative filtering algorithms implemented by Apache’s Mahout library (<http://mahout.apache.org>):

- *User-based collaborative filtering* [15] identifies a requester’s neighbors (the users with similar tastes) and uses their ratings and the level of similarity with the requester to compute a prediction of the requester’s ratings for the items she does not know yet.
- *Cluster-based collaborative filtering* [79] pre-groups users into clusters of similar users (according to their ratings) and averages the ratings of all users within each cluster to compute a prediction of the requester’s ratings for unknown items. We specifically use hierarchical clustering.
- *Slope One* [41] is an item-based algorithm that leverages on the principle of “popularity differential,” that is, on how much one item is liked more than another. In order to predict the rating of an item, it considers information both from other items rated by the requester (and their ratings from other users) and from other users who rated the item (and their ratings to other items) .
- *Singular Value Decomposition* (SVD) [37] is a matrix factorization algorithm that computes ratings out of features automatically extracted from a known, incomplete user-item matrix. The matrix is decomposed into a user-feature, a feature-item, and a feature-feature matrix. Rating predictions are computed as the product of the requester’s row, the feature-feature matrix, and the item’s column.

These algorithms have been selected as they are popular and simple, two properties that allow us to communicate better the effects of the data on



recommendation quality. Other algorithms have been shown to perform similarly or even better under certain conditions, but our goal is more that of understanding and communicating the effect with widely known and adopted algorithms. Since all restaurants in our dataset are easily reachable by foot, user location and time (the usual contextual information) are not needed; we consider instead the purpose the requester is interested in.

### 5.3.3 Quality metric

We compare algorithms based on their *precision* (since we don't have full knowledge of the users' interests - they may have rated only a subset of the restaurants they actually know - we cannot compute meaningful recall values). Given a user  $u$ , the *list* of computed recommendations, and the purpose  $p$ , we compare the performance of the algorithms using the following precision metric (following [23]):

$$Precision(u, p, list) = \frac{||Good(u, p, list)||}{||Good(u, p, list)|| + ||Bad(u, p, list)||}$$

where

$Good(u, p, list)$  = items in list that have been rated positively by user  $u$  for purpose  $p$ ,

$Bad(u, p, list)$  = items in list that have been rated negatively by user  $u$  for purpose  $p$ .

For the comparison, we split the users' ratings for each purpose  $p$  into a training set (the ratings the algorithms can use to build the user profile) and a test set (the ratings used to compute the precision of recommendations) with a 70/30 proportion. We tested only users that had at least 6 ratings per selected purpose, leaving at least 2 ratings for testing (the ceiling of the 30% split), and omitted items the users didn't express any opinion for. Test

ratings were randomly collected half from users' positive ratings and half from their neutral or negative ratings (to test good and bad predictions). To make the test independent of the computed split of ratings, each query was repeated with 5 different random splits.

### 5.3.4 Algorithms tuning and configuration

Given this evaluation strategy, all algorithms underwent a dry run to configure them for best performance: we collected 5 recommendations (i.e.  $Np = 5$ ) for each purpose from each algorithm and averaged the precision of each list of recommendations (20 per user: 4 purposes by 5 training/test splits). *Slope One* has no parameters, so it was used as it is. *User-based collaborative filtering* depends on the used similarity metric, neighborhood strategy and neighborhood size. We tested Pearson correlation, log likelihood, Spearman correlation, Tanimoto coefficient, cosine similarity, Euclidean-distance-based similarity and Yule similarity. The best precision was obtained with neighborhood selected by similarity threshold, using Yule similarity and similarity threshold 0.3, with a precision of 76%. For *cluster-based collaborative filtering*, we used the same similarity metrics as for user-based collaborative filtering and identified the best configuration in log likelihood similarity and stopping condition expressed as fixed number of clusters, set to 3, with a precision of 70%. The best precision with SVD was obtained with 10 features and 30 iterations, with 65% of precision.

## 5.4 Results

### 5.4.1 Aggregate precision

For the comparison, we select the top  $Np$  recommendations from TripAdvisor in the order proposed by TripAdvisor. We vary  $Np$  from 2 to 15 to study the effect of the recommendation set size on precision and compute the precision of our recommender algorithms by averaging the results of all the purposes over 570 individual data points (114 users times 5 random splits per run) per purpose.

For a first assessment of the difference between the dataset underlying TripAdvisor and our own dataset, we compare the recommendations of TripAdvisor with a similar non-personalized, average-based recommendation algorithm (we call this the baseline algorithm) using our dataset. This baseline algorithm computes the predictions of user ratings by computing the “lower bound of Wilson’s score confidence interval” (<http://www.evanmiller.org/how-not-to-sort-by-average-rating.html>). This formula computes a confidence interval for the average rating we would obtain if we had all ratings by the full population, starting from a sample of ratings. The lower bound tells “the item is liked at least that much.”

Our own dataset differs from TripAdvisor’s one in four key aspects: (i) 3-value vs. 5-value rating scales, (ii) purpose-based vs. generic ratings, (iii) locals vs. tourists, and (iv) small amount vs. large amount of ratings. Since we don’t have access to the actual dataset and algorithm used by TripAdvisor, we cannot distinguish the effects of each of these aspects, but we can still see in Figure 1 how TripAdvisor generally produces better recommendations than the baseline (except for  $Np=5$ ). The key to this better performance most likely lies in the bigger amount of ratings TripAdvisor can rely on.

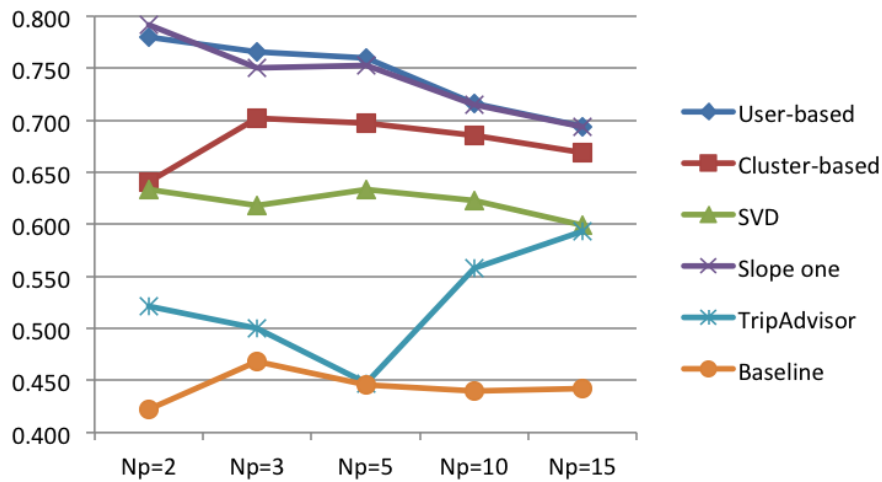


Figure 5.1: Precision of the recommendation algorithms for varying  $Np$ .

Interestingly, if we now look at the precision of the personalized algorithms, we see that they all perform better than both TripAdvisor and the baseline. Slope one and user-based have the best precision and are very close to each other. Cluster-based is not far from the top recommenders, with only a distance of 2 percentage points in precision for higher  $Np$ , while SVD performs worse. TripAdvisor's precision is higher for  $Np = 15$ , where it reaches the same precision of SVD, while it still is 10 percentage points lower than the best performance. This shows that as the size of the recommendation set grows, TripAdvisor has higher probability to contain good recommendations. In order to assess the expressive power of the charts in Figure 5.1, we took the precision values for  $Np = 15$  and performed pair-wise t-tests. The tests confirm also statistically what is communicated by the chart visually: except for user-based/slope one and TripAdvisor/SVD, all precision values are significantly different (p-value < 0.0001,  $\alpha$ -level = 0.05, considering the precision of 1280 recommendation lists for each algorithm).

Overall, Figure 5.1 shows that the precision of the best algorithm between the chosen personalized algorithms (user-based collaborative filter-

ing) is from 10 ( $Np=15$ ) to 31 ( $Np=5$ ) percentage points higher than that of TripAdvisor (from 17% to 68% in relative terms). This makes us accept our hypothesis H1: *data collected from locals and state-of-the-art, personalized recommendation algorithms produce recommendations of higher precision than TripAdvisor.*

This means that even though our dataset is significantly smaller than that of TripAdvisor, the focus on locals and personalization yield recommendations that are of significantly higher quality compared to recommendations computed with a generic algorithm from a much larger dataset. TripAdvisor’s restaurant rank is in fact built using a huge amount of reviews mostly by tourists and specifically focuses on recommending restaurants to tourists. Our experiment aims to understand how to recommend restaurants to locals and shows that locals are a special class of users that are simply more demanding than generic tourists.

We have to keep in mind that these results have been obtained by averaging the precision of purpose-based recommendations. TripAdvisor starts with a disadvantage since it is built only for making recommendations to tourists and its recommendations could be worse for the other purposes (as we will see in details below).

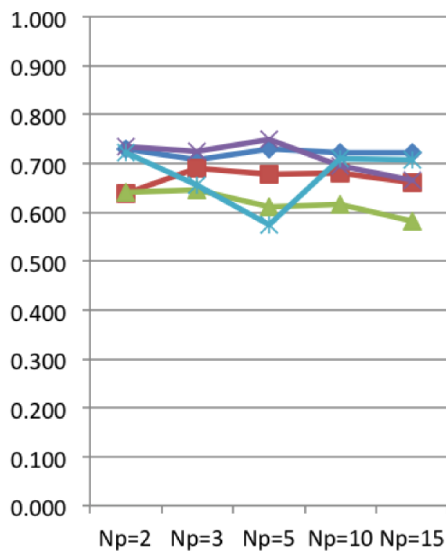
### 5.4.2 Purpose-specific precision

In order to understand this different demand, we now analyze the importance of purpose-specific ratings in recommending restaurants. In Chapter 4 we found that the restaurants perceived as good for bringing a tourist are similar to those for a romantic dinner with the partner, while the ones for going out with friends are very different and more related to the price/quality ratio. Next, we analyze concretely how the different recommenders behave depending on the purpose a user has in mind. The test setting of the experiments is the same as above, with the only difference

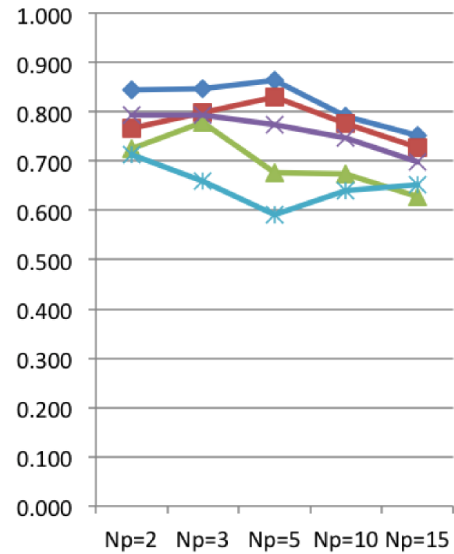
that now we no longer aggregate results and instead keep purposes separated.

Figure 5.2 reports the precision graphs for each purpose. If we concentrate on TripAdvisor, we see that it provides good predictions for a dinner with tourists, while its precision decreases if the meal is to be consumed with the partner or friends, and it reaches its lowest value if a good price/quality ratio is the target (only 26% of precision for  $Np = 2$ ). The personalized algorithms seem less affected by the purpose, with slightly higher precision for a dinner with the partner and slightly lower precision for the price/quality ratio. Slope one, user-based and cluster-based collaborative filtering always outperform SVD.

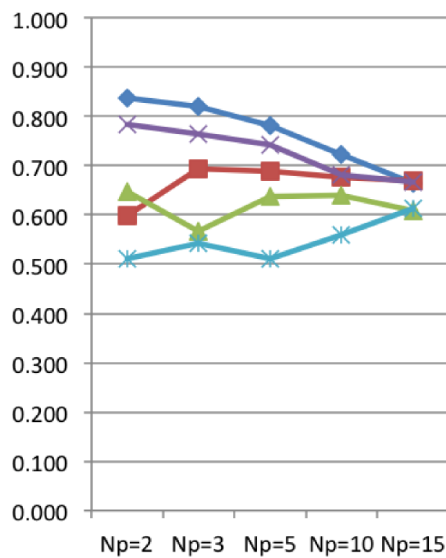
These results clearly indicate that each purpose is different from the others, and algorithms that take care of these differences are able to build better recommendations than generic algorithms. TripAdvisor shows the best precision for  $Np=2$  and “dinner with tourists”, while the worst precision is obtained for  $Np=2$  and “price/quality ratio”, with a difference of 46 absolute percentage points. Purpose-based recommender algorithms have a more constant quality, with less difference between the best and the worst precision: for example, user-based collaborative filtering has the highest precision for  $Np=5$  and “dinner with the partner”, while the lowest one is for  $Np=15$  and “price/quality ratio”, with a difference of 25 absolute percentage points. This minor difference demonstrates a higher quality of purpose-based, personalized recommendations under all circumstances. This supports hypothesis H2 for the purposes dinner with partner, dinner with friends and price/quality ratio: *recommendations computed from purpose-specific data outperform TripAdvisor* for these purposes and may represent a strategic value for competitors of TripAdvisor that want to target locals instead of generic tourists.



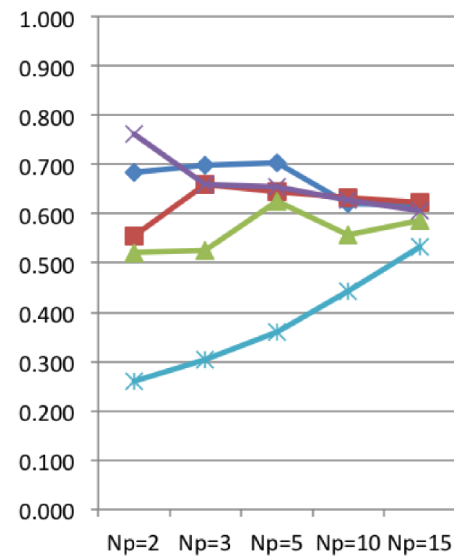
a) Precision for "dinner with tourists"



b) Precision for "dinner with the partner"



c) Precision for "dinner with friends"



d) Precision for "price/quality ratio"

Legend: User-based (blue diamond), Cluster-based (red square), SVD (green triangle), Slope one (purple cross), TripAdvisor (cyan asterisk)

Figure 5.2: Purpose-based precision for the five recommendation algorithms.

TripAdvisor recommendations have instead a high precision for a dinner with tourists, and for this purpose their quality is in line with the ones computed with personalized recommendation algorithms. Given that these latter algorithms use data that stem from locals, this means that locals essentially agree with TripAdvisor on where to bring a tourist and where not. This, in turn, is a quality certificate for TripAdvisor for this specific purpose.

## 5.5 Discussion and Conclusion

Our experiments show that providing locals with restaurant recommendations is a tricky endeavor, because providing them with added value compared to generic tourist portals like TripAdvisor asks for advanced personalization, not only based on identity but also on purpose. The experiments further show that if data are collected with an eye on the purpose of the restaurant visit and from locals, even basic algorithms outperform generic recommendations. The improvement in recommendation quality thanks to tailored data is not only significant, but has a big effect size. These results are somewhat surprising, given that also more advanced and precise algorithms are available in the literature. The results however also show that TripAdvisor is still competitive in its own domain, i.e., recommendations for tourists.

At first glance, our comparison of the personalized algorithms with purpose-specific datasets and TripAdvisor may not seem fair: TripAdvisor does not specifically target locals; the available datasets are very different; and its underlying algorithms are not publically available and known. However, at the same time TripAdvisor is one of the key representatives of the state of the art in restaurant recommendations and one that nicely shows a one-size-fits-all approach that works for tourists. What we show



in this chapter is that there is still huge space for improvement and businesses if the focus is shifted from a generic audience to locals. Yet, doing so really requires a thorough planning of how to collect the necessary data and how to tailor it to the needs of locals. There is no shortcut solution to good data collection (e.g., crawling TripAdvisor or similar), and each new application will have to collect its own data according to its specific needs.

The results of our experiments also reveal another, slightly hidden message that is of particular importance to the world of mobile recommender systems: Mobile devices have typically small screens and are often used in situations in which the user cannot pay full attention to the device. This means that the user can see only few recommended items at a time and may not be willing or able to go through a long list of recommendations [58]. A mobile recommender system is thus particularly challenged even more than a desktop one to compute precise recommendations. The data in Figure 5.2 show that TripAdvisor performs particularly weakly for small result sets. The lesson is that simply porting a desktop version of a recommendation algorithm to a mobile recommender system may be dangerous, and personalization and data quality become even more important.

One limitation of the study is that our comparison of algorithms is based on the externally visible behavior of TripAdvisor. Its actual, internal algorithm and dataset are not made publicly available. Further, the algorithms we used were trained on the same dataset we also used for testing. TripAdvisor, on the one hand, could rely on a much bigger set of ratings for the training, but, on the other hand, the comparison was again based on our dataset of ratings. One difference between the two datasets is that TripAdvisor uses a 5-star rating scale, while our dataset uses a 3-value thumbs-up/thumbs-down scale. Understanding if the two rating scales affected our findings would require further analysis.



# Chapter 6

## Designing Recommendations for Mobile Devices

### 6.1 Introduction

In the literature, many recommenders have been proposed and studied, with application mainly in e-commerce, movies or music, and mostly based on desktop interfaces [43, 7].

Restaurants have different characteristics that must be considered when building a recommender service. First, *geolocation* is important, as people can only experience places that are close to them. Second, people generally rate only a *limited amount of items*: while people can listen to many songs each week, they can eat only in few different restaurants. Third, people are interested in the *quality* of restaurants and the *experience* they can have there. This type of information is not available through restaurant characteristics, but can be collected through other people's opinions.

TripAdvisor ([www.tripadvisor.com](http://www.tripadvisor.com)), Yelp ([www.yelp.com](http://www.yelp.com)), and Foursquare ([www.foursquare.com](http://www.foursquare.com)) are among the most widely used restaurant recommenders. In particular, Foursquare recently turned from a location-based social network into a recommender service for restaurants, bars and other places for leisure activities. The primary use of these recommender

systems is via mobile devices, also since decisions on restaurants are often taken on the go.

Recommenders that provide their services through mobile devices have to consider the specific aspects of these devices. These are not only related to *limited screen real estate* and the specifics of *touch-based input*, but also the fact that interaction is often performed with *limited attention*, e.g., due to users performing other activities in parallel, such as walking or talking. Yet, the fact that people carry their mobile phone always with them also provides for new opportunities, e.g., to collect ratings: proactively asking people for ratings right when they leave a restaurant (thanks to geo-localization) and asking them to review the experience they just had can provide for better contextualized and precise ratings. Doing so however asks for rating interfaces that make this action really fast even in contexts of split attention as the user is likely to walk at that time.

In this chapter, we study which combination of rating scale and recommender algorithm provides the best results for mobile restaurant recommendations, with a specific focus on ratings entered on mobile devices and in conditions of limited attention. We start from identifying which *rating scales* are most effective for collecting user inputs through mobile devices. We study effectiveness under three dimensions: (i) the ability of users to quickly insert ratings on the go and in conditions of limited attention, which are common conditions when interacting with a mobile device; (ii) the feeling of accuracy and completeness that users perceive for the various rating scales; and (iii) whether the different rating scales impact the quality of the results we can get from recommender systems. We then analyze ratings (and the effectiveness of the different rating scales) with different algorithms to identify which *recommender algorithm* has highest precision and user satisfaction. We specifically focus on collaborative filtering algorithms [37], given their ability to capture other people's opinions.

Finally, we compare the best combination of rating scale and algorithm against *popular recommendation services*.

So far, similar analyses have been conducted only for desktop interfaces, and mobile devices have not been studied adequately. In fact, most studies on mobile restaurant recommenders have focused on how to integrate contextual information, such as location, weather and time of the day [5, 77], and paid less attention to rating scales and algorithms used to compute personalized recommendations. Also, recommendations are predominantly tested by simulating user requests using a training dataset and checking whether recommended items are liked in a testing dataset. This offline evaluation fails to capture how people perceive the recommendations and how satisfied they are with them [12]. The research presented in this chapter considers both *offline* and *user-centric* evaluations of algorithms to identify the best one to use in a mobile restaurant recommender.

## 6.2 Rating on Mobile Devices

Recommendation services are based on the collection of ratings to learn user tastes. This is done using a rating scale. Fixed the number of options to choose from, the rating scale must be represented visually. For example, the same 5-value rating scale can be represented with stars like in Amazon or with dots like in TripAdvisor. Different representations may influence users' choices of values [3]. Mobile devices are often used while performing other activities, reducing the attention the user can dedicate to the device. Given this limited attention, rating scales should be adequate to the limited screen real estate and the specifics of touch-based input.

In this section we study the first two dimensions of rating scales announced in the introduction: (i) the ability of users to quickly insert ratings on the go and in conditions of limited attention, which are common condi-

tions when interacting with a mobile device, and (ii) the feeling of accuracy and completeness that users perceive for the various rating scales.

### 6.2.1 Data collection

Since in the mobile context it is important to keep the cognitive load of the rating task low, we expect that thumbs-up/thumbs-down is preferred over the 5-star scale (previous works showed that on desktop devices the difference between the two is very small [68]). Thanks to past studies, we also know that the neutral rating plays a fundamental role [20]: it does not force the user to decide when she has not a clear opinion about an item. For this reason, we think that a 3-value rating scale would perform better than thumbs-up/thumbs-down. Moreover, having less options to choose from, we expect it to require less time and effort than 5-star. To test the effect of different representations, we include in our study two different 3-value rating scales: 3-thumb, which adds the neutral thumb to thumbs-up/thumbs-down, and 3-face, in which rating values are expressed with sad, neutral and happy faces. This leads us to 4 rating scales to evaluate (Figure 6.1):

- *2-thumb*, where users can select thumbs-up if they like the item or thumbs-down if they do not like it, without any neutral option;
- *3-thumb*, similar to 2-thumb but with the intermediate thumbs-neutral for a neutral rating;
- *3-face*, another 3-valued rating scale with “like” represented with a smily face, “neutral” represented with a neutral face, and “don’t like” represented with a sad face; and
- *5-star*, where users can select from 1 to 5 stars, with 5 stars being the maximum and no half stars allowed.

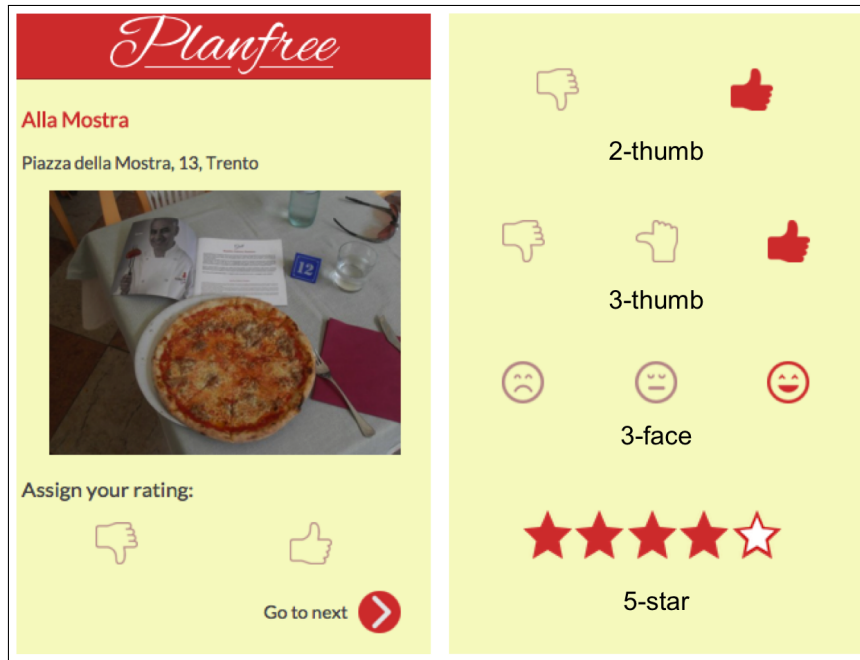


Figure 6.1: The UI for collecting ratings: the actual task on the left; the rating scales dynamically selected on the right.

Table 6.1: Number of participants per rating-scale (columns) and context condition (rows).

	2-thumb	3-thumb	3-face	5-star	total
standing	19	20	19	20	78
walking	15	14	19	19	67
total	34	34	38	39	145

Next, we consider two contexts of mobile device usage:

- *standing* (or sitting): when people can devote all their attention to the device and are not doing other activities in parallel; and
- *walking*: when their attention is split between the device and walking and the environment around them.

We collected ratings for 50 restaurants in the city center of Trento, Italy, measured how many seconds pass between the request of a restaurant

rating and the saving of the rating, and questioned participants about their satisfaction with the rating scale they used. To do so, participants were asked to access the study website through their smartphones, see restaurants in random order once at a time (Figure 6.1), and rate them according to the rating scale randomly assigned to them.

To enroll participants, in October 2015 we invited students at our university to participate in our study by rewarding them with a free coffee. This allowed us to attract 145 participants. Some of them were randomly selected for the walking context, in which they were asked to walk while rating restaurants. Participants had the possibility to skip restaurants they did not know, and the rating task was stopped after 10 ratings or after viewing at most 35 of the 50 restaurants in our list. In this way, we hide part of the restaurants that the participants may know, which will be needed for the next user studies on recommender algorithms. We collected a total of 1295 ratings, with an average of 9 ratings per participant. Table 6.1 shows how participants were divided by rating scale and by context condition.

### 6.2.2 Rating efficiency

We analyzed how much time was required to select a rating in function of the different rating scales and context conditions. For each rating, we measured how many seconds passed between the request of a rating and the actual rating. We expected that ratings assigned while walking and rating scales showing more options, such as 5-star, would require more time. The actual results are shown in Table 6.2.

The lowest and the highest average times are obtained in the standing context: 3-thumb is the rating scale that requires less average time, but 5-star, the slowest scale, is less than 1 second slower. In the walking context, the tendency of smaller scales to be fastest is not confirmed as 5-star obtains



Table 6.2: Statistical analysis of time needed (in seconds) to select a rating under different context conditions.

	Mean	Standard Dev.
standing * 2-thumb	5.937	3.469
standing * 3-thumb	5.848	2.133
standing * 3-face	6.183	2.108
standing * 5-star	6.807	2.757
walking * 2-thumb	5.971	1.557
walking * 3-thumb	6.631	2.052
walking * 3-face	5.971	1.959
walking * 5-star	5.946	1.971

the lowest average time, while 3-thumb got the highest, being 0.7 seconds slower. These differences are not only small, but they are not statistically significant according to ANOVA analysis (rating scales:  $p - value = 0.88$ ,  $\alpha - level = 0.05$ ; context:  $p - value = 0.81$ ,  $\alpha - level = 0.05$ ). Given the high standard deviations, we expect that also with more participants the differences would not become statistically significant. What is more surprising is that there is almost no difference in task performance between the two usage conditions (standing vs. walking).

Despite our expectations, it seems that the different rating scales and contexts have only a very limited effect on the rating performance and that there is no single rating scale that gives particular advantages over the others. This result could be explained by the fact that the task is relatively easy, and people that use smartphones every day, like university students, have no difficulty in performing it even in contexts that require them to split their attention.

### 6.2.3 Questionnaire

After the rating assignment, participants were asked to assess the rating scales they used also qualitatively by stating their agreement with the following 6 statements:

1. *The options available to express my opinion require a suitable amount of time for choosing.*
2. *The rating options are in correct number.*
3. *The rating options let me express my opinion with the correct accuracy/expressiveness.*
4. *A lot of precision is needed to select the correct rating.*
5. *The rating options and their visualization are suitable to my context.*
6. *I am satisfied with the available options to express my opinion.*

Participants expressed their agreement with the statements via a 5-point Likert scale from strongly disagree (-2) to strongly agree (+2). Participants were either Italian or English-speaking foreign people, so the sentences were available in both languages and the one matching the device language was shown. The average agreements per statement and rating scale are reported in Table 6.3.

3-face obtained the highest agreement regarding the adequacy of the *time* needed to rate restaurants, closely followed by 5-star (with a difference of only 0.08). This difference is very small and not confirmed by the times we measured. That is, the subjective perception of 5-star is better than its actual performance.

Statements 2 and 3 analyzed the *granularity* of the rating scales. Both collected a very high agreement for 5-star, with a difference of 0.48 from

Table 6.3: Average agreement per questionnaire statement

Statement	2-thumb	3-thumb	3-face	5-star
1	1.012	0.918	<b>1.132</b>	1.047
2	0.339	0.539	0.500	<b>1.024</b>
3	0.014	-0.032	0.184	<b>0.539</b>
4	-0.195	-0.139	<b>-0.316</b>	0.024
5	0.544	0.386	0.632	<b>0.793</b>
6	0.098	0.243	0.447	<b>0.792</b>
Sum	2.202	2.193	3.211	<b>4.172</b>

3-thumb for sentence 2 and a difference of 0.36 from 3-face for sentence 3. These results confirm that people have a preference for rating scales with different levels of granularity that allow them to better articulate their opinions. For statement 2, all the differences between rating scales involving 5-star results statistically significant ( $\alpha$  - level: 0.05;  $p$  - values: 0.002 for 2-thumb, 0.02 for 3-thumb, 0.01 for 3-face), while the differences between the other three rating scales are not significant. For statement 3 we got almost the same results, but here the difference between 5-star and 3-face is not significant ( $\alpha$  - level: 0.05;  $p$  - values: 0.03 for 2-thumb, 0.04 for 3-thumb, but 0.11 for 3-face).

The *precision* needed to select a rating is analyzed with sentence 4, for which low agreement expresses better quality. 3-face obtained the lowest agreement, followed by 2-thumb and 3-thumb. As expected, 5-star resulted the worse, being the only one with somehow positive agreement with the sentence, indicating that the precision needed to select one of the stars is higher than selecting one of the thumbs or faces.

Regarding the *context* condition (standing vs. walking), only 2-thumb received higher satisfaction when walking, while the other rating scales are preferred when standing. The more comfortable rating scale is 5-star,

Table 6.4: Details of statement 5: effect of context conditions

Context	2-thumb	3-thumb	3-face	5-star
standing	0.421	0.700	0.842	<b>0.850</b>
walking	0.667	0.071	0.421	<b>0.737</b>
difference	0.246	0.629	0.421	<b>0.113</b>
average	0.544	0.386	0.632	<b>0.793</b>

with the higher satisfaction both for standing and for walking and with the lowest difference between the two contexts (Table 6.4).

Finally, the last sentence studied *satisfaction*. 5-star stands out as the rating scale with the highest satisfaction, followed by 3-face, 3-thumb, and 2-thumb. The differences between 5-star and both 3-thumb and 2-thumb are statistically significant ( $\alpha$  – level: 0.05;  $p$  – values: 0.01 for 3-thumb, 0.005 for 2-thumb); for 3-face it is not.

If we sum up all agreement levels per rating scale (after multiplying the agreement with statement 4 by -1 to turn the preference for the lowest value to the preference for the highest one as in the other statements), 5-star clearly obtained the highest overall agreement. Pairwise t-tests between these results confirm that the differences between 5-star and both 2-thumb and 3-thumb are statistically significant ( $p$  – values: 0.0005 and 0.001;  $\alpha$  – level: 0.05). The difference between 5-star and 3-face did not reach statistical significance.

#### 6.2.4 Discussion

These results support our intuition that in mobile devices the 5-star rating scale requires more time or effort to choose a rating representing the user’s opinion due to the higher granularity. On the other hand, this more effort needed is small, not confirmed by the actual time needed to assign a rating, and seems to be accepted by users as a reasonable cost for the

higher granularity with different nuances. In this case the user evaluation is fundamental and without it we would have reached the wrong conclusion considering all rating scales equally adoptable as they all need almost the same time to assign a rating. 5-star has been recognized as the best also for desktop devices [68, 10, 22], so the same rating scale can be used in both situations.

One possible explanation of the preference of 5-star over the other rating scales is the omnipresence of this rating scale: people are so used to it that they can immediately recognize it and they perfectly know how to use it. The 3-thumb rating scale may be affected by the opposite problem: people are used to thumbs, but the presence of the neutral option may make people feel like there is something wrong with it.

Surprisingly, 3-face obtained a higher acceptance than 3-thumb (with only one exception: for sentence 2 3-thumb obtained a higher satisfaction, but the difference is of only 0.04). Even though both rating scales have the same number of options with the same meanings, the different representation influenced users' satisfaction. Maybe the fact that 3-face is completely new makes the presence of the neutral value less strange. On the other hand, the distribution of ratings between negative, neutral and positive is almost the same in both cases (21 negative ratings, 117 neutral and 187 positive for 3-thumb, and 36 negative ratings, 119 neutral and 199 positive for 3-face), so the different representation did not bias the feedbacks towards any extreme of the rating scale.

### **6.3 Personalized Recommender Algorithms**

Given a dataset of ratings collected using a specific rating scale, the next step is identifying which algorithm builds the best recommendations with the provided ratings.

In the following, we present our evaluation of collaborative filtering algorithms, first offline with an automatic evaluation tool, and then user-based, where participants to the first study were contacted again to collect their feedback on the recommendation lists computed with the selected algorithms.

### 6.3.1 Offline evaluation

#### Method

The first study allowed us to collect a good dataset of restaurant ratings (1295 ratings for 50 restaurants, 145 users); each user provided 3 to 10 ratings. Previously we have seen that, when recommending restaurants, recommendation algorithms should take into consideration also geographic distance from the requester: only places that are relatively close to her position should be recommended. To avoid adapting the algorithms we study in the following (to take geolocation into account), we limited the items in the dataset to restaurants in a restricted geographical area: the selected restaurants are the restaurants that can easily be reached by walking inside the city center of Trento, Italy.

To study which recommendation algorithm performs best with which rating scale, we selected a set of off-the-shelf collaborative filtering algorithms:

- User-based collaborative filtering [15]
- Hierarchical cluster-based collaborative filtering [34]
- K-means cluster-based collaborative filtering [19]
- Slope one [41]
- Singular Value Decomposition (SVD) [37]

Since cluster-based collaborative filtering is highly dependent on the clustering algorithm used, we decided to test it with two different clustering algorithms: for *hierarchical clustering* we selected a complete-linkage bottom-up approach that builds high quality clusters but is not scalable, while *k-means* is a lighter algorithm that is very fast and scalable but can produce low quality clusters.

For the implementation, we used the Apache Mahout library (`mahout.apache.org`) written in Java that contains many off-the-shelf personalized recommender algorithms and an evaluation tool that we adapted to our needs. The evaluator takes one user at a time and splits her own ratings into a 70% training set (known to the algorithm) to learn her tastes and a 30% test set (not known to the algorithm) to evaluate the computed recommendations. With 3-10 ratings per user, this provides each algorithm with 2-7 ratings to learn from. All the ratings added by the other users are used in the computation of the recommendations (i.e., they are part of the training set). To make the evaluation independent of the specific split of ratings, we used five different random splits of ratings into training and test sets, computing recommendations of five restaurants five times for each user.

The quality of the produced recommendations is measured through the  $F_{0.5}$  measure, which is based on *precision* and *recall*. These formulas are based on the number of *true positives* (*tp*) (items that are recommended and have been positively rated by the user in the test set), *false positives* (*fp*) (items that are recommended but have been rated as neutral or negative by the user in the test set), and *false negatives* (*fn*) (items that are not recommended but have been rated positively by the user in the test set). Items that are recommended but have not been rated by the user are ignored and are not considered in the computation of precision and recall.

The three metrics are defined as follows:

$$precision = \frac{tp}{tp + fp} \qquad recall = \frac{tp}{tp + fn}$$

$$F_{0.5} \text{ measure} = (1 + (0.5)^2) * \frac{precision * recall}{((0.5)^2 * precision) + recall}$$

*Precision* thus indicates how many items in the recommendation list are actually liked by the user, while *recall* indicates how many of the items the user likes have been included in the recommendation list.

Given the peculiarities of mobile devices (small screen real estate, only few recommendations visualizable at a time, limited attention by users), precision is more important than recall. For this reason, we selected the  $F_{0.5}$  measure as indicator of recommendation quality instead of the more common  $F_1$  measure. While the  $F_1$  measure computes the harmonic mean of *precision* and *recall*, the different weights used in  $F_{0.5}$  measure give higher importance to *precision*.

## Results

Before studying which algorithm performs best with which rating scale, we identified the best parameters for each algorithm for each rating scale. Slope one has no parameters, so it was used as is, while cluster-based collaborative filtering with k-means clustering was almost constantly failing to recommend any of the items known by the user, so it was not possible to evaluate its results. For this reason, we decided to apply here the same parameters (similarity metric and number of clusters) as for hierarchical cluster-based collaborative filtering. Without being able to evaluate the quality of the clusters built with this specific algorithm, this represents the best approximation of parameters possible. User-based collaborative filtering and cluster-based collaborative filtering depend on a *similarity metric* to compute the similarity between users to identify neighbors and



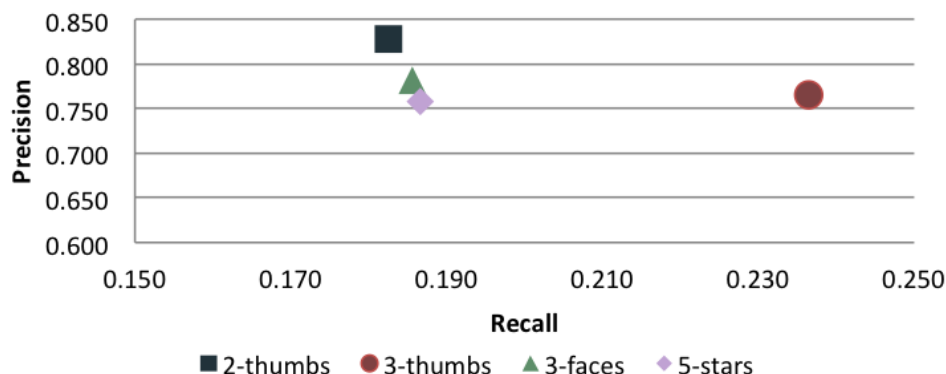


Figure 6.2: *Precision/recall* of best algorithm per rating scale.

cluster-mates. In both cases we tested the following measures: cosine vector similarity, Pearson correlation similarity, Euclidean-distance-based similarity, Tanimoto coefficient similarity, generalized Jaccard similarity (an extension of Jaccard similarity that can be applied also to rating scales with higher granularity than binary ratings), Yule similarity, log likelihood similarity, and Spearman correlation similarity. User-based collaborative filtering was tested with varying neighborhood sizes, while cluster-based collaborative filtering was tested with both fixed numbers of clusters and threshold-based stopping conditions. SVD was tested with different numbers of features and iterations.

This parameter tuning process allowed us to identify first the best configuration for each algorithm and then the best algorithm for each rating scale. The results are as follows:

- *2-thumb*: cluster-based collaborative filtering with hierarchical clustering, generalized Jaccard similarity and 8 clusters;
- *3-thumb*: user-based collaborative filtering with cosine vector similarity and neighborhood of 10 users;
- *3-face*: cluster-based collaborative filtering with hierarchical clustering, Tanimoto coefficient similarity and 2 clusters;

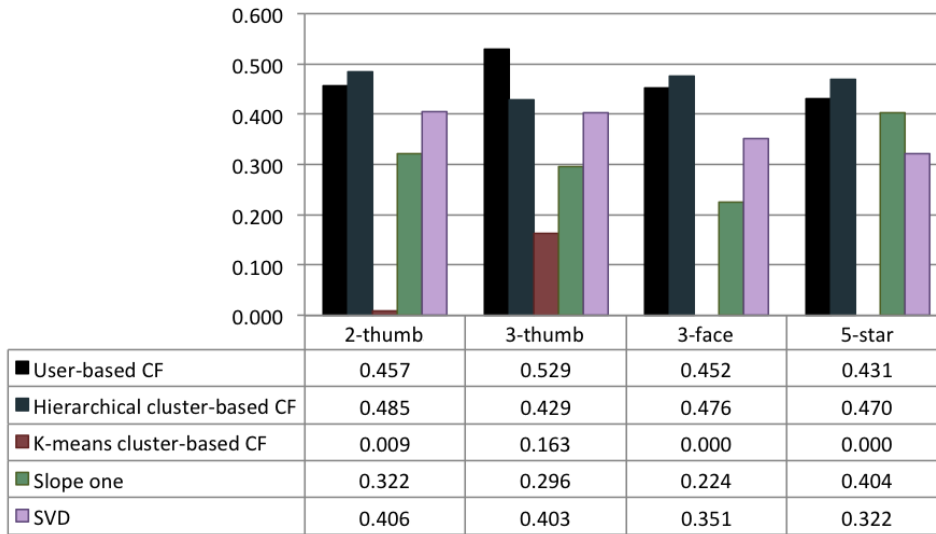


Figure 6.3: Searching the best algorithm for each rating scale,  $F_{0.5}$  measure.

- *5-star*: cluster-based collaborative filtering with hierarchical clustering, log likelihood similarity and 2 clusters.

Figure 6.2 illustrates the *precision* and *recall* values computed for each rating scale using the identified algorithms and configurations. The resulting  $F_{0.5}$  measure of each algorithm and rating scale are reported in Figure 6.3. For three out of four rating scales the best algorithm is cluster-based collaborative filtering with hierarchical clustering. User-based collaborative filtering performs best for 3-thumb ratings. Interestingly, all the scale/algorithm combinations perform similarly well, while the combination 3-thumb/user-based collaborative filtering have the highest recall, resulting in 9% higher  $F_{0.5}$  measure.

### 6.3.2 User evaluation

#### Method

As a follow-up of the previous study on the rating scales, we contacted the same participants by email and asked them to evaluate the recommenda-

Table 6.5: Number of participants per rating scale.

2-thumb	3-thumb	3-face	5-star	total
11	5	10	15	41



Figure 6.4: UI for recommendation list assessment.

tions computed with the five algorithms identified in the offline evaluation. Out of 145 initial participants, 41 participated also in this second experiment. Table 6.5 shows how many participants per rating scale responded to our invitation.

Each participant was presented with 5 different personalized recommendation lists (one for each algorithm) in random order. Each list contained 5 restaurant recommendations, and participants were able to indicate whether they agreed with a recommendation (they would have rec-

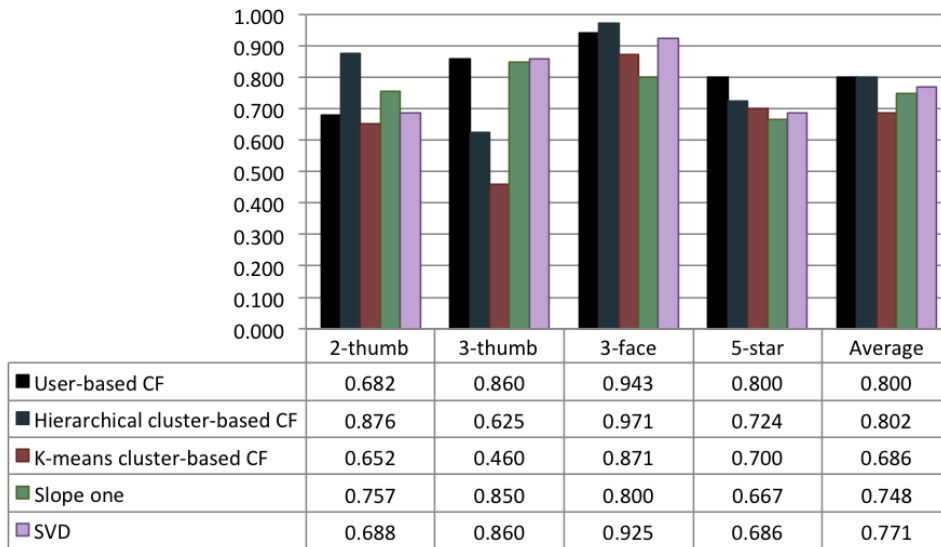


Figure 6.5: *Precision* of recommendations according to users.

ommended that restaurant too) or not (they wouldn't). For restaurants they did not know they were not required to enter any feedback. The recommended lists showed only places the participants did not already rate in the data collection phase (first study); since that phase omitted some well known restaurants, participants were able to identify among the recommended restaurants some known restaurants they did not rate before. Participants were also asked to express the overall satisfaction with each complete list of recommendations. Satisfaction was expressed using a Likert scale with 5 degrees ranging from “not satisfied at all” to “very satisfied”, mapped to numeric values from -2 to +2. The interface used for the study is shown in Figure 6.4.

The design of this study allows us to compute a *subjective* precision: the recommended items a user agrees with are considered *true positives*, the ones she disagrees with are considered *false positives*.

## Results

Figure 6.5 summarizes the collected data regarding the *precision* of the algorithms for each rating scale and the average precision (weighted by the number of participants for each rating scale). The highest precision is obtained by cluster-based collaborative filtering with hierarchical clustering and user-based collaborative filtering, while the lowest is obtained with cluster-based collaborative filtering based on k-means clustering (with a relative reduction in average precision of 14%).

If we look at the single rating scales, we can see that the precision for 3-face ratings is very high. It is always the highest, except for slope one, where 3-thumb received a higher precision.

The participants knew only some of the recommended items, so we were able to collect only partial precision of the algorithms. The precision results we reported here does *not* indicate a difference between the algorithms of statistical significance. These results are still valuable as they confirm the ones we already obtained with the offline evaluation: user-based collaborative filtering and hierarchical, cluster-based collaborative filtering are the two algorithms providing the best recommendations.

Figure 6.6 reports the data collected regarding the *satisfaction* with the individual recommendation lists. Looking at the average overall satisfaction (weighted by the number of participants per rating scale), the highest satisfaction is obtained by user-based collaborative filtering, followed by hierarchical cluster-based collaborative filtering with 29% lower satisfaction. The lowest satisfaction is obtained by slope one, with a reduction of 68%. Still all algorithms received an average positive satisfaction, which means that the participants were generally either neutral or positively satisfied with the recommendations. Only 15 negative satisfaction ratings were assigned, mainly to slope one and SVD, while the positive satisfaction ratings

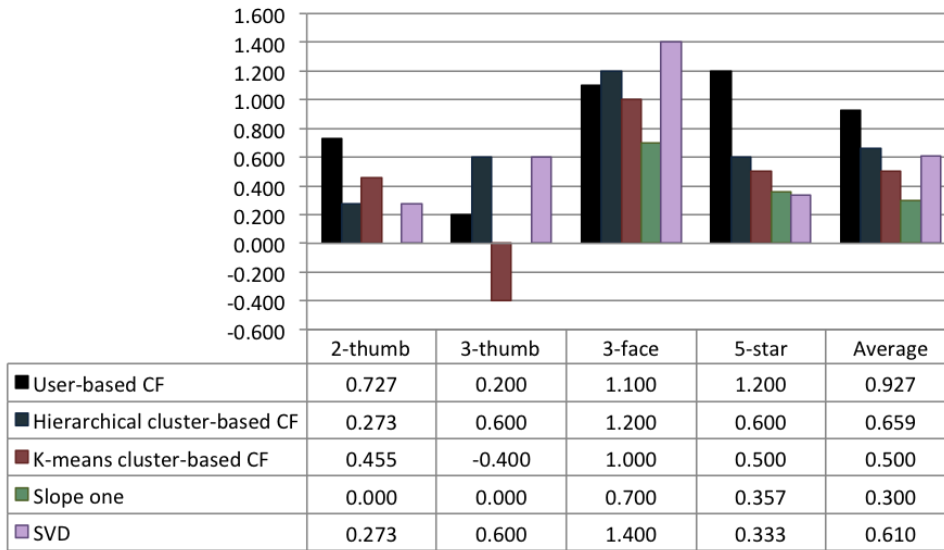


Figure 6.6: User satisfaction with recommendation lists.

were 103. One may wonder why k-means cluster-based collaborative filtering obtained the most negative satisfaction for 3-thumb, even though slope one and SVD received more negative satisfaction ratings. The value is justified by the overall low satisfaction and the low number of respondents for the 3-thumb rating scale (5): one negative (-2) and four neutral (0).

If we look at each rating scale, we can notice that the highest satisfaction is consistently obtained by 3-face ratings, except for user-based collaborative filtering where participants that rated restaurants using 5-star showed higher satisfaction. It is interesting to notice how much difference there is between the different rating scales for SVD: 3-face participants showed a satisfaction of 1.5 while all the other participants showed a satisfaction lower than 0.6.

An ANOVA test on the algorithms (without splitting the satisfaction results by rating scale) demonstrated that at least two algorithms are different in a statistically significant way ( $p - value: 0.04$ ,  $\alpha - level: 0.05$ ,  $F - critical: 2.42$ ,  $F: 2.53$ ). With t-tests between pairs of algorithms, we found that the differences between user-based collaborative filtering and

both k-means, cluster-based collaborative filtering and slope one are statistically significant. Participants were able to indicate their satisfaction for all the recommendation lists they evaluated, providing us with more data than for precision.

### 6.3.3 Discussion

Given the presented results, we can conclude that the best algorithm for our dataset is user-based collaborative filtering (due to higher user satisfaction and precision), immediately followed by hierarchical cluster-based collaborative filtering (which has the same precision).

Previous work identified SVD as the best algorithm for personalized recommendations, in particular for recommending movies [7, 40, 12]. In this case, instead, its recommendations are of lower quality and the algorithm scores only third best. Our dataset has a higher density than the movie dataset used in the other studies, and this could be the reason for this result: SVD is known to work better on low-density datasets, where user-based collaborative filtering does not have enough information to build good recommendations. We expect restaurant datasets to be more dense than movie datasets in general: the available options in an area are not so many as movies, and the most popular restaurants will receive ratings from almost every user.

It is interesting to note how the overall satisfaction level expressed by participants sometimes differs from the pure precision values reported. The satisfaction refers to the impression participants have of all five recommendations as a whole (see Figure 6.4) and seems only marginally related with the precision of each single recommended item in the list. Very likely, the collected satisfaction levels also include external knowledge of the participants, e.g., coming from recommendations received by friends or relatives or other kinds of advertisement they had been exposed to.

## 6.4 Comparison with Commercial Services

We now compare the quality of recommendations computed with user-based collaborative filtering and rating-scale-specific parameters with the quality of recommendations computed by two popular online restaurant recommenders: TripAdvisor and Foursquare. This allows us to understand how the recommender service based on the best algorithm we identified performs compared to services used in practice. TripAdvisor uses a 5-values scale represented with dots, while Foursquare allows 3 possible answers to the question “do you like this place?”: “yes”, “no” and “so-so”.

### 6.4.1 Offline evaluation

#### Method

We follow the same approach as in the previous offline study: we compute recommendation lists of 5 items and evaluate them using the  $F_{0.5}$  measure. We use the dataset of ratings collected in the first study (with 1295 ratings for 50 restaurants given by 145 users) and evaluate the algorithms on five 70/30 splits of the user ratings into training and test sets.

The algorithms used by TripAdvisor and Foursquare are not known, so we cannot run their recommendation algorithms on our dataset. We thus use the generic recommendations both services provide through their Web interface. Despite this limitation, this study is still interesting as it can give us an indication of how the service we are designing can work against its competitors. TripAdvisor and Foursquare are able to compute better recommendations, but also our service can be improved with novel, more accurate versions of the algorithm we selected.

TripAdvisor and Foursquare ranks for all restaurants considered for the evaluation of our own algorithms were extracted and saved at the time we collected ratings for our first study. For each user, we retrieved



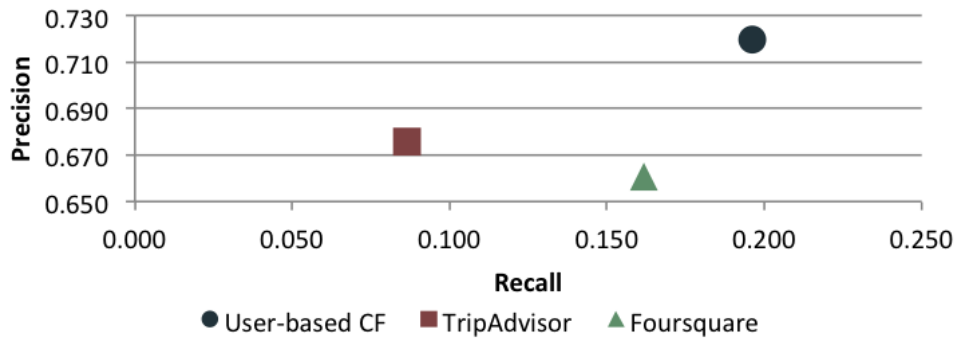


Figure 6.7: *Precision* and *recall* of offline evaluation of user-based collaborative filtering, TripAdvisor and Foursquare.

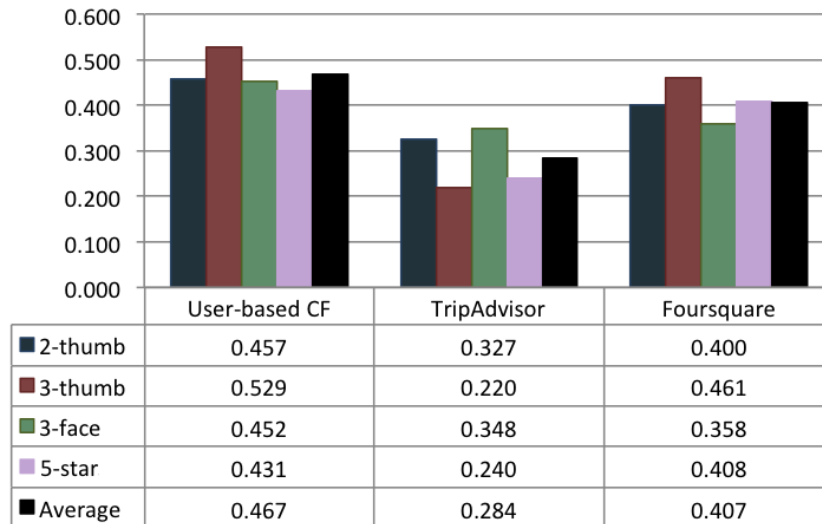


Figure 6.8:  $F_{0.5}$  measure of offline evaluation of user-based collaborative filtering, TripAdvisor and Foursquare.

the full, ranked list of recommendations as computed by TripAdvisor and Foursquare, removed the restaurants already rated by the user in the training set, and retained the top 5 restaurants according to the remaining rank.

## Results

Figure 6.8 reports the  $F_{0.5}$  measures obtained. User-based collaborative filtering and Foursquare are close, while TripAdvisor recommendations are clearly of lower quality (with a reduction of 13 percentage points in

Table 6.6: Number of participants per rating-scale.

2-thumb	3-thumb	3-face	5-star	Total
10	7	7	15	39

$F_{0.5}$  measure). Looking at the chart of *precision* and *recall* (Figure 6.7), we can see that the low recommendation quality of TripAdvisor is mainly due to a low recall, while the difference in precision is small.

The algorithms provide almost the same quality across the different rating scales, with only TripAdvisor showing significant differences. This result is surprising, since TripAdvisor is based on 5-star ratings, while Foursquare is based on 3-values ratings. Surprisingly TripAdvisor does not produce the best recommendations for our 5-star participants, but for the ones that used the 3-face rating scale.

## 6.4.2 User evaluation

### Method

We involved again the participants already enrolled in the first user study and asked them about their opinion regarding the recommendation lists computed for them by user-based collaborative filtering, TripAdvisor and Foursquare, respectively. We contacted again the participants by email, using the same user interface as in the previous experiment (Figure 6.4), but with the three new recommendation lists. Again, participants were able to indicate whether they agreed with recommendations individually and to express their satisfaction with the full list of recommendations. We collected the answers of 39 participants out of the 145 people contacted by email (Table 6.6).

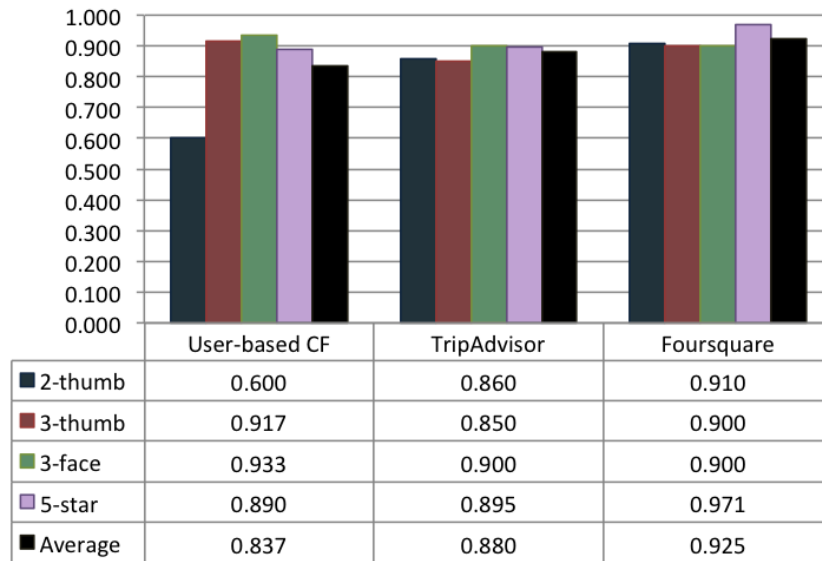


Figure 6.9: *Precision* of recommendations according to users.

## Results

12 participants did not know any of the items in the three recommendation lists, so we were not able to compute respective precision values. For the remaining participants, we obtained the *precision* levels shown in Figure 6.9. We can see that user-based collaborative filtering produced the lowest precision according to the participants, with 9% lower average precision than Foursquare, which reached the highest precision. TripAdvisor is very good too, with 5% lower average precision than Foursquare. Pair-wise t-tests were not able to reach statistical significance, due to the low number of participants.

Looking at the details per rating scale, we can see that surprisingly the precisions of TripAdvisor and Foursquare have only minor fluctuations across the different rating scales, while user-based collaborative filtering has difficulties in producing good recommendations for 2-thumb users. This could be due to the limited number of choices that does not allow the algorithm to clearly understand the different tastes of users.

Figure 6.10 reports users' satisfaction per rating scale and the average satisfaction (weighted by the number of participants per rating scale). The highest average satisfaction is obtained by user-based collaborative filtering, immediately followed by Foursquare, while TripAdvisor has a significantly lower satisfaction (with a reduction of 86%). An ANOVA test on the algorithms (without splitting the satisfaction results by rating scale) demonstrated that at least two algorithms are different in a statistically significant way ( $p$  - value: 0.003,  $\alpha$  - level: 0.05,  $F$  - critical: 3.07,  $F$ : 6.19). With t-tests between pairs of algorithms, we found that the differences between TripAdvisor and both user-based collaborative filtering and Foursquare are statistically significant ( $p$  - values: 0.0006 and 0.015,  $\alpha$  - level: 0.05), while there is no statistically significant difference between user-based collaborative filtering and Foursquare.

Taking a deeper look into the details for each rating scale, we surprisingly see that TripAdvisor got negative satisfaction for the 5-star rating scale, which is the one actually used by the service itself. While Foursquare generated almost the same satisfaction throughout all the rating scales, user-based collaborative filtering got very good results for 5-star and 3-face, while it performed poorly for 2-thumb and 3-thumb.

### 6.4.3 Discussion

One result that stands out is the big difference between precision and satisfaction of TripAdvisor according to our participants. According to participants, the recommendations made by TripAdvisor were quite precise, yet they left them not fully satisfied; the result was similar also in the offline evaluation of TripAdvisor. That is, participants acknowledge that the recommended restaurants are of good quality, but they simply would have preferred to see other restaurants in the top-5 selection. In

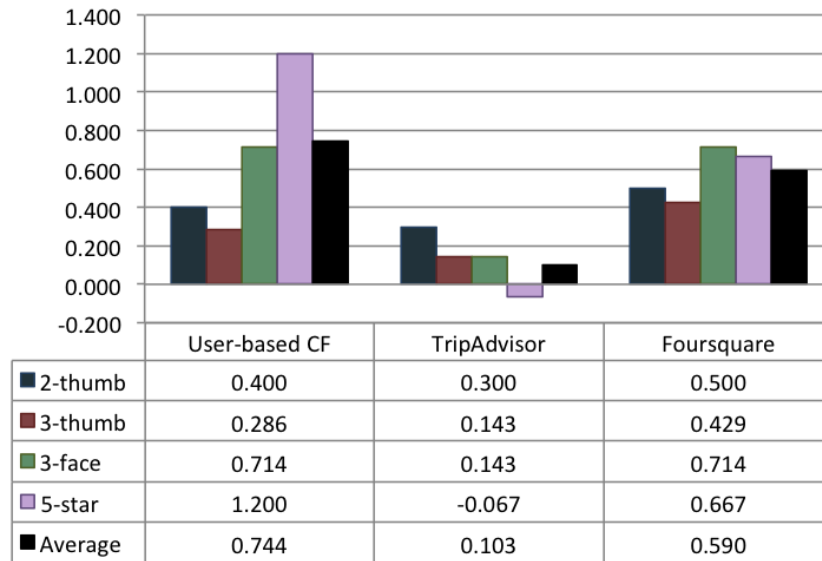


Figure 6.10: User satisfaction with recommendations lists.

short, this finding unveils the big shortcoming of TripAdvisor: the lack of personalization.

Another reason for the lower satisfaction generated by TripAdvisor is the different user-base compared to ours. TripAdvisor is generally dedicated to tourists, while Foursquare specifically focuses on locals (i.e., people that live in the city they are searching a restaurant for) and is popular among a younger audience (86% of users are younger than 44 years, <http://www.quora.com/What-are-the-demographics-of-Foursquare-users>). Our dataset is based on the opinions of university students living in Trento, an audience that is thus more similar to that of Foursquare than to the one of TripAdvisor. This focus of our dataset represents a limitation of our study.

Looking at the rating scale-based results of user-based collaborative filtering, we found another reason to prefer 5-star over the other rating scales. In fact, the personalized algorithm produces the best results when ratings are collected with this scale (obtaining the highest precision, both offline and user-based, and the highest satisfaction): the granularity is

higher than in the other rating scales and it is likely that this lets the algorithm better learn the different nuances of user tastes.

## 6.5 Learnings and Limitations

We studied the performance of different rating scales and recommender algorithms for mobile devices using offline, empirical analyses and user studies, specifically comparing *precision*, *recall* and  $F_{0.5}$  *measure*. According to the results reported on in this chapter, the combination of rating scale and recommender algorithm that suits best our restaurant recommendation problem is 5-star with user-based collaborative filtering. 3-face is considered by users slightly less accurate than 5-star, while cluster-based collaborative filtering with hierarchical clustering has a performance close to that of user-based collaborative filtering. The study shows well how the subjective opinion of users does not always follow objective metrics, e.g., the 5-star rating scale outperforms the other scales in terms of user preferences although we could not identify any objective difference.

We also compared user-based collaborative filtering with two of the most popular restaurant recommender systems, TripAdvisor and Foursquare, and found that TripAdvisor produces worse recommendations than the other two algorithms, very likely due to the lack of personalization and a different user-base. The TripAdvisor and Foursquare datasets we used were generic and did not consider individual user tastes, but they had the advantage of being much larger compared to our own dataset. Yet, we lack a direct comparison of the algorithms, as these services are businesses and do not share their data or algorithms. Still the comparison shows how already a simple personalized recommender algorithm with purposefully collected data can outperform generic recommendations of currently-used services – and this is what counts in practice.

One limitation of this study is the size of the datasets it is based on. Our dataset consists of 1295 ratings, while usually algorithms are evaluated on datasets of 100,000 ratings or more (however, most works in literature test their performance on the MovieLens dataset and not on real restaurant recommendations). As for the user evaluations in Sections 6.3 and 6.4, these are based on the answers provided by 40 participants; we expect that larger sample sizes would have made the differences between algorithms stronger and significant. Despite these limitations, we think that the described results clearly indicate a tendency of preference for the 5-star rating scale by our participants. Of course, so far we focused on university students, which are not necessarily representatives of the full population. However, this focus allowed us to unveil the shortcoming of TripAdvisor in terms of personalization.

Finally, it is important to note that user-based collaborative filtering and hierarchical cluster-based collaborative filtering perform well in offline settings with limited amounts of data, while they do not scale well and do not support online updates as users rate new items: user neighborhoods and clusters are computed offline, e.g., nightly or once a week. More work is needed toward fast and scalable, online algorithms, e.g., to quickly compute high-quality clusters even with large amounts of data as for instance proposed in [52] and [9]. The findings of this study justify the necessary effort.





# Chapter 7

## Collecting the Initial Ratings from the Crowd

### 7.1 Introduction

Providing precise and accurate recommendations requires first and foremost data, that is, user ratings of items. These data can be used i) for research purposes to study the behaviour of novel algorithms, ii) for development purposes to train and fine-tune the selected recommendation algorithm, and iii) to bootstrap a novel recommender system to make it able to build recommendations even for the first users accessing it. In these cases volunteering is not applicable as there is not a recommender system available, so data cannot be collected directly from the users using the system, consuming recommendations, and providing their own ratings and feedback. It is therefore needed to gain access to useful data without involving the users of the future system.

So far, we specifically enrolled people to collect their opinions and evaluate our ideas for recommender systems. We collected ratings through an online survey and advertised it at university and in the city center, we invited our friends and colleagues to participate and to spread the word, and assigned some rewards to attract higher participation. Despite our

advertisement and rewards it was hard to collect enough participants, and we started to explore different ways for collecting datasets of ratings.

One way of achieving this that has gained momentum over the last years is *crowdsourcing*, i.e., outsourcing of a piece of work (e.g., the rating of places) to an unknown group of people via an open call for contributions [30]. Crowdsourcing has two key pros with respect to online surveys: i) the survey can be advertised to many “workers” around the world, and ii) respondents can be motivated offering rewards (typically small amounts of money).

The rewards are great because they let researchers collect a huge amount of data at relatively low cost and in short time. On the other hand, rewards motivate respondents to cheat, providing random or wrong answers, to complete the task as fast as possible, increasing their personal hourly income. Moreover, to build a good-quality dataset of ratings we need techniques to control that the ratings are reliable (i.e. people are not cheating) and well-distributed across the items. In fact, if some items remain without ratings, the algorithm will not be able to recommend them.

It would thus be beneficial to build a rating platform that takes into consideration rating quality and sparsity. The platform should ideally acquire the best ratings possible from the currently active crowd of workers based on their expertise with respect to the items. The system should automatically identify cheaters, i.e. workers that are consistently providing misleading ratings. We distinguish two types of cheaters: i) *lazy workers*, who assign ratings randomly to complete the rating tasks as fast or as effortless as possible, and ii) *malign workers*, who give misleading ratings to particular items in order to reduce or raise their average ratings. The platform should also prioritize acquiring ratings for underexposed items (i.e., items that have fewer ratings).

While traditional crowdsourcing platforms such as Amazon Mechanical Turk ([www.mturk.com](http://www.mturk.com)) and CrowdFlower ([www.crowdflower.com](http://www.crowdflower.com)) have been successfully used in various scenarios to acquire human knowledge in a cheap and effective manner, they are not suitable for our scenario. In such platforms, there is no clear notion of a worker expertise apart from a single score reflecting how well a worker performed on previous tasks. This is not feasible in our scenario for various reasons. First, to be able to measure how well workers performed on previous tasks (rating of items in our case), some ground truth (i.e., correct ratings for items) must be available. Ratings are subjective and worker-dependent as they reflect the personal experience of the workers and their personal tastes. This in turn means that there is no single ground truth that can be used to compute expertise scores for workers.

Second, workers might be more skilled to rate certain types of items than others. For instance, students might be more knowledgeable about holes-in-the-walls or cheap restaurants whereas professionals might be more knowledgeable about fancier restaurants. It is thus crucial to associate workers with different scores representing their expertise with respect to the different types of items being rated.

Finally, it is crucial to ask workers to rate only the items for which they have higher skills. In traditional crowdsourcing platforms, workers self-appoint themselves to tasks and requesters have no control over how the task assignment is carried out. In our case, we would like the platform to automatically assign tasks to workers based on their estimated expertise.

In this chapter, we present a novel crowdsourcing platform that acquires reliable ratings for a set of items from a set of workers. A reliable rating is a truthful rating provided by an *expert* worker. Our platform estimates worker expertise based on the agreement of the worker with other *similar expert* workers in the system. The platform makes use of a fine-grained

utility function to present workers with the best items to rate based on the workers' expertise and the number of ratings the items have. Finally, the framework automatically identifies cheaters and we experiment with various ways of dealing with them. Our platform is described in Section 7.3.

We evaluated our framework using a set of exhaustive experiments on both real and synthetic datasets about restaurant. Our experimental results, presented in Section 7.4, clearly highlight the effectiveness of our system in acquiring reliable ratings from expert workers, particularly for underexposed items. We also show that identifying cheaters, which is an integral part of our platform, has a great impact on the overall rating quality.

According to our knowledge, this is the first work that addresses the issue of acquiring reliable *ratings* from the crowd. Most related work studied how to estimate the skills of crowdsourcing workers assuming the existence of only one valid ground truth [13, 31, 33, 78, 25], which is not the case in our setting as we have already explained. Moreover, most related methods for estimating worker skills are generally post-processing methods, so they are not applicable in our scenario where we make use of the worker skills during task assignment to improve rating quality as more tasks are being performed. This is also not the case for existing work where one-time pre-task qualification tests are run to estimate worker skills.

Our main contributions are the followings:

- We build a scalable and realistic crowdsourcing platform to acquire reliable ratings of items such as restaurants, movies or hotels.
- Our platform automatically estimates worker expertise based on the agreement of the worker with similar expert workers in the system.

- Our platform uses a carefully designed utility function to present workers with the best items to rate based on the estimated expertise of the workers and the number of ratings for those items.
- Our platform automatically identifies cheating workers and can dampen their effect.

## 7.2 Problem Definition

Given a set of workers  $W$  and a set of items  $I$ , our goal is data acquisition. That is, we want to acquire *reliable* ratings for as *many* items as possible where the set of possible ratings  $R$  is  $\{0$  (don't know),  $1$  (don't like),  $3$  (neutral),  $5$  (like) $\}$ . We map ratings to values between 1 and 5 to stay compatible with the 5-star rating paradigm used by most recommender systems.

More specifically, we want to populate a database of tuples of the form  $T = \langle w, i, r \rangle$ , where  $w$  is a worker,  $i$  is an item, and  $r$  is the rating provided by  $w$  for item  $i$ . Our data acquisition has the following two sub-goals: 1) worker  $w$  should not be a cheater, and 2) item  $i$  should currently be the best item for worker  $w$  to rate, meaning that  $i$  is the item  $w$  is most knowledgeable about and that has the fewest ratings (the platform parameters give more importance to one or the other aspect).

Note that the first sub-goal, identifying cheaters, is important for the realization of our main goal, acquiring reliable ratings for items. In the second sub-goal, given that  $w$  is not a cheater, we want her to rate the items that have fewer ratings she is most knowledgeable about and for which she is most likely to give reliable ratings. In the next section, we describe our framework that realizes the above sub-goals to achieve our main goal of acquiring reliable ratings for as many items as possible from a crowd of workers.

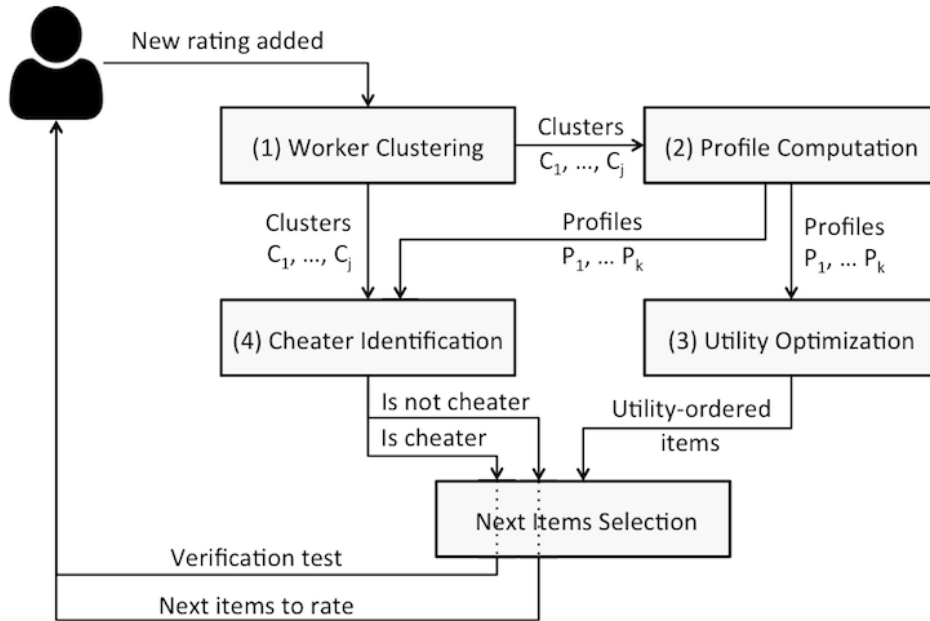


Figure 7.1: The proposed framework.

### 7.3 Framework

In a nutshell, our framework works as follows. First, we cluster the items to be rated into  $n$  itemsets  $I_1, \dots, I_n$  according to characteristics of the items. The clusters can overlap and the characteristics are inherent properties of the items. For example, in the case of restaurants, the characteristics can be cuisine, price range, etc. We cluster items into itemsets so that we can associate each worker in our system with an expertise level for each itemset that can help us assess how likely the worker can provide reliable ratings for items in an itemset. A given worker can be an expert in one type of items and less expert or completely unknowledgeable for other types. Second, we constantly cluster active workers in the platform based on the ratings they provide: each cluster represents a group of workers with same tastes, different from the other ones. Clusters are used in our system for two reasons: 1) to compute the expertise of workers, and 2) to identify workers who provide misleading ratings which we refer to as *cheaters*.

A diagram of our framework is shown in Figure 7.1. Our platform works in task sessions: the worker enters a new task session with the first ratings she assigns and continues to rate the items the platform presents her till she leaves the session. Within a session, the reliability of her ratings is checked. Once a new worker joins the system, she is asked to rate the  $l$  items with the highest number of ratings in the system so far, to be able to compare her with as many workers in the system as possible. Based on her provided ratings, we assign the worker into one worker cluster. After a worker is assigned to a cluster, her profile is updated for each itemset she provided ratings for. Using the calculated profile, the worker is either suspected to be a cheater and is asked to pass a verification test, or she is asked to rate more items. A verification test is simply another set of rating tasks where a worker is asked to rate items she has rated before. A worker passes the verification test if she is relatively consistent with her previous ratings, otherwise she is banned from the system. The verification test is designed this way to disguise it from actual cheaters and to not turn off falsely-flagged workers. Other ways of dealing with cheaters are also applicable and we experiment with different methods in Section 7.4.

In case more items are to be rated (i.e., the worker was not flagged as a cheater or has passed the verification test), we apply a utility function that presents the worker with the items we anticipate that she has the most promise to rate reliably, prioritizing items that have fewer ratings. Our framework performs the above mentioned procedure for every worker that is currently active in the platform and the same procedure is repeated each time a new rating is provided until the worker leaves the system. In case a worker consistently fails to join a worker cluster after she has provided a given number of ratings, the worker is suspected to be a cheater and is asked to pass a verification test as before.

Our framework consists of four main components: 1) Worker Clustering, 2) Profile Computation, 3) Utility Optimization, and 4) Cheater Identification. We describe each component next.

### 7.3.1 Worker clustering

Given the set of active workers  $W$  in the system, our goal is to cluster them into a number of clusters based on their ratings. Let  $w$  and  $w'$  be two workers in  $W$  and  $I_{int} = \langle i_1, i_2, \dots, i_n \rangle$  be the set of items that both  $w$  and  $w'$  rated. Also, let  $R = \langle r_1, r_2, \dots, r_n \rangle$  be the vector of ratings worker  $w$  provided for the items in  $I_{int}$  where  $r_k$  is the rating of item  $i_k$ . Similarly, let  $R' = \langle r'_1, r'_2, \dots, r'_n \rangle$  be the vector of ratings worker  $w'$  provided for the items in  $I_{int}$  where  $r'_k$  is the rating of item  $i_k$ . We cluster the workers based on an adjusted Euclidean-distance-based similarity measure which is computed as follows:

$$sim(w, w') = 2 * \left( \frac{1}{1 + \frac{\sum_{k=1}^n (r_k - r'_k)^2}{n(r_{max} - r_{min})^2}} - \frac{1}{2} \right)$$

where  $r_{max}$  is the highest rating possible (5 in our case) and  $r_{min}$  is the lowest rating possible (1 in our case). In the computation of similarity we are ignoring the “don’t know” ratings with value 0.

Note that using a standard Euclidean-distance-based similarity or a Cosine Vector similarity will not work in our setting, where the neutral rating (3) should be considered very similar to both “like” (5) and “don’t like” (1). In standard similarity metrics, instead, rating 3 is considered quite different from both 1 and 5. To overcome this, we started from the standard Euclidean-distance-based similarity metric

$$EuclideanDistanceSim(w, w') = \frac{1}{1 + \frac{\sqrt{\sum_{k=1}^n (r_k - r'_k)^2}}{\sqrt{n}}}$$



and substituted the simple distance ( $\sqrt{\sum_{k=1}^n (r_k - r'_k)^2}$ ) with a relative distance by dividing it by the maximum distance there could be between any two ratings (in our case 4). This results in a value in the range of  $[-1, 1]$  and squaring it gives a value in  $[0, 1]$  which is a diminished distance to accommodate for the closeness between the neutral rating and the other two ratings. Finally, the whole fraction gives a result in the range of  $[0.5, 1]$  (after calculating the average, adding 1 and then taking the reciprocal). Thus, we subtract  $1/2$  and multiply by 2 to map the results back to the interval  $[0, 1]$ . Of course, any other suitable similarity measure can be seamlessly used instead in our framework depending on the context for which the framework is used.

Our incremental clustering algorithm is shown as Algorithm 1. The algorithm is called whenever a new rating is provided, since it is a new evidence about what the worker knows and the goal of the clustering is to group together workers who have similar tastes and experiences. This is also the case when a new worker joins the system and there already exists a set of worker clusters. We utilize an *incremental hierarchical* clustering algorithm [34]. We opted for a hierarchical clustering rather than a flat one since the number of clusters is not known a priori and it changes over time as workers rate more items or as new workers join the platform. Hierarchical clustering is also best suited for incremental clustering as it avoids recomputing the full hierarchy of clusters each time a new rating arrives. Note that we do not store the full hierarchy of clusters at the end, but only store the final level ending with a flat set of clusters.

Our clustering algorithm takes as input 1) the worker who provided the new rating (or a new worker) which we refer to as the provoking worker  $w$ , 2) the current set of worker clusters  $H$ , and 3) a cluster closeness threshold  $\tau_C$ , and it returns a new set of clusters. In case the provoking worker  $w$  was an existing worker who has rated a new item, we remove  $w$  from its

**Algorithm 1** Cluster Workers

---

**Input:** current set of clusters  $H$ , provoking worker  $w$ , closeness threshold  $\tau_C$ **Output:** new set of clusters

```
if  $w$  is an existing worker then
   $C_w \leftarrow getCluster(H, w)$ 
   $removeWorker(C_w, w)$ 
  if  $C_w$  is empty then
     $removeCluster(H, C_w)$ 
  end if
end if
 $new \leftarrow createNewCluster(w)$ 
 $addCluster(H, new)$ 
while TRUE do
   $max \leftarrow -\infty; first \leftarrow null; second \leftarrow null$ 
  for  $i = 1$  to  $|H| - 1$  do
    for  $j = i + 1$  to  $|H|$  do
       $closeness \leftarrow closeness(C_i, C_j)$ 
      if  $closeness > max$  then
         $max \leftarrow closeness$ 
         $first \leftarrow C_i; second \leftarrow C_j$ 
      end if
    end for
  end for
  if  $max > \tau_C$  then
     $merged \leftarrow mergeClusters(first, second)$ 
     $replaceClusters(H, first, second, merged)$ 
  else
    break
  end if
end while
return H
```

---

own cluster and assign it to a singleton cluster. In case  $w$  is a new worker who has just joined the system, she is also assigned to a singleton cluster after providing a predefined number of ratings for the most rated items.

Our clustering algorithm then keeps on merging the two closest clusters to reduce the number of clusters by one at each iteration, following a bottom-up approach.

The closeness of two clusters  $C_i$  and  $C_j$  is computed as the smallest similarity between their workers (i.e., complete-linkage clustering) as follows:

$$closeness(C_i, C_j) = \min_{w \in C_i, w' \in C_j} sim(w, w')$$

Finally, we merge the two clusters  $C_i$  and  $C_j$  with the *highest* closeness. We keep on merging clusters as long as the following condition holds:

$$\exists_{C_i, C_j} closeness(C_i, C_j) > \tau_C$$

where  $\tau_C$  is a threshold on the closeness between any two clusters to be merged.

We use complete linkage to ensure that when we remove the provoking worker from her cluster, the intra-cluster similarity either stays the same or increases. This in turn means that the affected cluster stays compact, does not need to be split to improve the clustering quality, and we can start improving clusters from the current configuration of clusters, without recomputing the complete clustering hierarchy. Since we only have one more cluster at each step, very few iterations are needed for updating the clusters and this is independent from the number of clusters and workers involved.

### 7.3.2 Profile computation

Each worker  $w$  is associated with a profile vector  $\langle w.p_1, \dots, w.p_n \rangle$  representing the worker's expertise for each itemset  $I_j$ , where  $w.p_j$  is the ordered pair  $(w.p_j.known, w.p_j.skill)$ . To compute this profile, we measure two aspects of the worker: how many items in  $I_j$  she knows (i.e., did not rate 0), which we refer to as  $w.p_j.known$  and, for the items she knows, how

much she agrees with other expert workers from her cluster, which we refer to as  $w.p_j.skill$ . The first component of the worker profile  $w.p_j.known$  is computed as follows:

$$w.p_j.known = \frac{\#known(w, I_j)}{\#ratings(w, I_j)}$$

where  $known(w, I_j)$  is the number of items that worker  $w$  knows in  $I_j$  (i.e., did not rate as 0) and  $\#ratings(w, I_j)$  is the number of items in  $I_j$  she has rated so far (including the 0 rating).

The second component measures how skillful the worker is for the items she knows. For the skill component, we utilize the agreement of the worker  $w$  with other workers from her cluster. The intuition behind this is that the worker is expected to behave similarly to the rest of the workers in her cluster. Before we dwell into the details of how we compute agreement between workers, we need to decide on who to compute agreement with. One alternative is to compute the agreement of worker  $w$  with all other workers from her cluster. This is however prone to some fundamental issues. First, in case some non-expert workers are still present in the current worker cluster, their effect on the agreement might deteriorate the profile values computed for other, possibly, expert workers. Moreover, if we compute the agreement with all the workers in the cluster of  $w$ , we would need a lot more ratings to have sufficient enough ratings to compute agreements between workers. Note that agreement depends solely on ratings provided by workers for items in the current itemset of interest. This is a problem since we ideally would like to acquire minimum number of ratings from non-expert workers. In order to overcome the aforementioned issues, we propose the following. To compute the skill value  $w.p_j.skill$  for worker  $w$ , we measure the agreement of worker  $w$  with only the top- $k$  most expert workers for the itemset  $I_j$ . We explain how to retrieve the top- $k$  most expert workers in a given cluster later.

Regardless of whether we measure agreement with all workers in a cluster or with only expert workers, the rest of the computation procedure for the skill component of a worker’s profile is the same. To measure agreement between two workers  $w_i$  and  $w_j$ , we use the same similarity metric used for building our clusters described in the previous subsection. Our similarity metric is well adapted to our setting of ratings and is applicable even when only few items have been rated by both users.

Once we have the agreement of the worker  $w_i$  with all the top- $k$  expert workers in her cluster, we aggregate the agreements by taking the *average* and use this as the skill for worker  $w_i$  on itemset  $I_j$  as follows:

$$w.p_j.skill = \frac{\sum_{w' \in top-k} agreement(w, w')}{k}$$

where top- $k$  is the top- $k$  most expert workers in  $w$ ’s cluster.

**Retrieving the top- $k$  most expert workers in a cluster.** Recall that in order to compute the skill component of the profile of a worker  $w$ , we need to measure the agreement between  $w$  and the top- $k$  most expert workers in her cluster  $C_w$  with respect to an itemset  $I_j$ . In order to retrieve these top- $k$  most expert workers, we rank all the workers  $w \in C_w$  in decreasing order of their skill components  $w.p_j.skill$ . We then take the top- $k$  workers with the highest  $w.p_j.skill$  values. Ties are broken arbitrarily using  $\#known(w, I_j)$  which is the number of items that the worker knows (i.e., did not rate as 0) from itemset  $I_j$ .

Initially, we bootstrap the system with a set of experts, for instance, restaurant or movie critics. These initial experts are clustered based on their ratings and their skills are computed based on the overall agreement between them. At step  $n$ , when worker skills need to be updated, the top- $k$  most expert workers are selected based on their skills computed at step  $n - 1$  and those are used to update the worker skills.

### 7.3.3 Utility optimization

The goal of the utility optimization component is to pick the best item for a given worker  $w$  to rate. More precisely, we want to pick the items with few ratings the worker most likely knows and will be able to reliably rate. To be able to do this, we use a utility function that is composed of two sub-components. The first component,  $SetUtility(w, I_j)$ , takes into consideration the worker profile and the number of ratings the worker has already provided for the itemset  $I_j$ . The second component,  $ItemUtility(w, i)$ , takes into consideration the number of ratings available for the item  $i$  and the closeness of the item to other items the worker knows.

More precisely, given a worker  $w$  and an itemset  $I_j$ , the  $SetUtility$  component is defined as follows:

$$\begin{aligned} SetUtility(w, I_j) &= \beta_1 \cdot \left(1 - \frac{\#ratings(w, I_j)}{MAX_k \#ratings(w, I_k)}\right) \\ &+ \beta_2 \cdot (w.p_j.known * w.p_j.skill) \end{aligned}$$

where  $\beta_1 + \beta_2 = 1$ ,  $\#ratings(w, I_j)$  is the number of ratings the worker  $w$  provided for  $I_j$  and  $w.p_j$  is the profile value of worker  $w$  for  $I_j$ . The first component of the  $SetUtility$  favors itemsets for which the worker has provided fewer ratings. The second component measures how expert the worker is with respect to the itemset.

Similarly, given a worker  $w$  and an item  $i$ , the  $ItemUtility$  component is defined as follows:

$$\begin{aligned} ItemUtility(w, i) &= \beta_3 \cdot \left(1 - \frac{\#ratings(i)}{MAX_j \#ratings(j)}\right) \\ &+ \beta_4 \cdot \frac{\sum_{j \in I_k^w} sim(i, j)}{|I_k^w|} \end{aligned}$$

where  $\beta_3 + \beta_4 = 1$ ,  $\#ratings(i)$  is the total number of ratings for item  $i$  and  $I_k^w$  is the set of items that worker  $w$  knows (i.e., has not rated as 0).

The similarity  $sim(i, j)$  is the similarity between two items  $i$  and  $j$  and it can be measured based on characteristics of the items (e.g. geographic distance between restaurants).

The final utility function  $utility(w, i)$  of item  $i$  belonging to itemset  $I_j$  for worker  $w$  is then computed as the average of  $ItemUtility(w, i)$  and  $SetUtility(w, I_j)$  as follows:

$$utility(w, i) = \frac{ItemsetUtility(w, I_j) + ItemUtility(w, i)}{2}$$

Note that the utility of item  $i$  for worker  $w$  or  $utility(w, i)$  is equal to 0 if worker  $w$  has already rated item  $i$  since we do not want to acquire more than one rating for an item by the same worker.

Once the utilities of every item for a given worker  $w$  are computed, we pick the item  $i$  for which  $utility(w, i)$  is maximum and provide this item to the worker  $w$  to rate.

### 7.3.4 Cheaters identification

One constant goal of our framework is to identify cheaters, that is, workers who are consistently providing misleading ratings. We distinguish two types of cheaters: i) *lazy workers*, which assign ratings randomly to complete the rating tasks as fast or as effortless as possible, and ii) *malign workers*, which provide misleading ratings to particular items in order to reduce or raise their average ratings. Our platform makes use of its different components to achieve this task. Given a threshold  $\tau_S$ , the worker  $w$  is considered a cheater if the following condition holds:

$$\forall_j w.p_j.skill \leq \tau_S$$

In addition, a worker  $w$  is considered a cheater if she consistently remains in a singleton cluster after  $m$  number of ratings have been collected (other than don't know or 0). In either case, the flagged worker is asked to pass

a verification test by asking her to rate items she previously rated. We then measure the agreement between the new ratings and the old ratings, and if the agreement is below a threshold value  $\tau$ , the worker is verified to be a cheater and is banned from the system. Otherwise, the worker profile is updated based on the agreement between the worker's new and old ratings. In the next section, we experiment with other strategies to deal with cheaters such as weighting down their ratings when aggregating items' ratings.

## 7.4 Evaluation

We evaluate the effectiveness of our framework for acquiring reliable ratings from expert workers using four different sets of experiments. The first set verifies the quality of ratings acquired for a real dataset of restaurants by assessing the performance of a recommendation system after identifying cheaters. This set of experiments clearly highlights the importance of the identification of cheaters. In these experiments, we also test the effect of worker expertise with respect to itemsets on the quality of the ratings acquired.

Next we perform parameter tuning to study the effect of the different parameters in our system such as the clustering algorithm parameters, cheaters identification threshold and the weights used in the utility function. Parameter tuning was performed on both synthetic and real datasets about restaurants.

The third set of experiments studies our utility function more closely and compares it with a number of alternative utility functions to test its effect on the overall performance of the system. In all these three experiments, we used the case of lazy workers to represent cheaters, as it was easier to simulate and since the results of the experiments hold regardless of the



type of cheaters. In the fourth and final experiment, we focus on the other type of cheaters, namely malign workers, which are workers who intentionally give misleading ratings to particular items in order to reduce or raise their average ratings. In particular, we evaluate the effectiveness of our framework in identifying such cheaters.

### 7.4.1 Rating quality experiments

**Effect of Filtering Out Lazy Workers.** The main goal of our work is to build a crowdsourcing service for collecting reliable high-quality ratings. One major application that could benefit from our work is recommendation. Our basic assumption is that acquiring reliable ratings will in turn improve recommendation accuracy. To validate this hypothesis, we use a real dataset of restaurant ratings, identify lazy workers in this dataset, and measure the errors made by an off-the-shelf recommendation system. More precisely, we measure the errors made when using all the ratings and when only using the ratings of trusted workers, i.e. workers that were not identified as lazy workers by our framework.

To build our real dataset, we collected ratings for 50 selected restaurants in Grenoble, France, from students and researchers using a custom website. We had a total of 57 workers, seven of which were experts and 10 were lazy workers and acquired a total of 540 ratings. The set of experts was composed of people that were very familiar with the restaurants in the selected city. On the other hand, the set of lazy workers was composed of workers who provided random ratings. In the end of this section, we experiment with the other type of cheaters, i.e. malign workers.

Since we want to understand the quality of the ratings we collected, we evaluate the recommendations built using subsets of our ratings to identify which subset is of higher quality and let the algorithm build better recommendations. We used a user-based collaborative filtering as a recom-

mendation algorithm [40], and cosine similarity on rating vectors to define, for each user, a fixed-size neighborhood of 10 most similar users. Such a configuration has been shown to perform quite well and is very popular in many successful recommendation systems [40].

We ran a recommender algorithm evaluation based on a 70-30 training-test split of data and root mean squared error (RMSE) as evaluation metric. We ran the algorithm using the training set as known ratings and predicted the ratings (computed as similarity-weighted mean of neighbors ratings) for the worker-item pairs already present in the test set. In this way, we compared the predicted rating and the real rating assigned by the worker to the item and measured the error (according to RMSE) the algorithm made. The smaller the error the better the prediction, and hence the better the quality of the data in the training dataset. For evaluation, we considered only “known” ratings (i.e. ratings 1, 3 and 5) and we split the dataset by time, identifying a specific date such that 70% of the ratings in our dataset were provided before that date (i.e. the training set) and 30% of the ratings were provided after (i.e. the test set).

Using the full dataset, we obtained an RMSE of *2.202*. Removing lazy workers, the RMSE was *1.021*. We can therefore conclude that the ability to isolate cheaters in this dataset reduced recommendation error by *53.6%*. This result is quite promising and shows the utility of cheater identification for a popular recommendation algorithm.

**Effect of Filtering Out Ratings of Non-expert Workers.** Moreover, we know which itemsets the workers are deemed to be more experts for, and we can exploit this information to filter out lower quality ratings (i.e., those for which workers are not considered to be experts enough to rate). Recall that our utility function makes use of the worker profile  $w.p_j$  to identify the itemsets for which the worker can give the best ratings. This means that ratings provided to items in itemsets for which the worker has higher

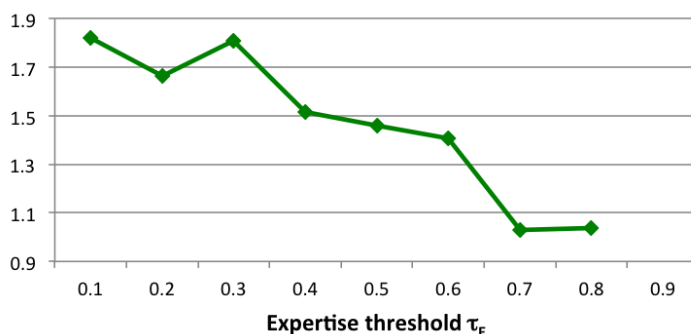


Figure 7.2: RMSE as ratings for itemsets on which workers are less expert about are filtered out.

profile values should be of higher quality since the worker is considered to be an expert for items they contain. For this reason, by filtering out the ratings workers provided to itemsets they are less experts for, i.e., with lower profile value, we will increase the quality of the ratings used to compute recommendations. Recall that the profile value of a worker  $w$  for an itemset  $I_j$  is composed of two components: 1)  $w.p_j.skil$  which is measured as the agreement of worker  $w$  with the top- $k$  most experts in her cluster with respect to itemset  $I_j$ , and 2)  $w.p_j.known$  which is measured as the number of items in  $I_j$  the worker knows (i.e., did not rate as 0). Finally, the two components of the worker profile are then used in our utility function to represent the worker expertise as  $expertise(w, I_j) = w.p_j.known * w.p_j.skil$ .

The minimum expertise value we obtained in our dataset was zero, so we tried different expertise thresholds  $\tau_E$  ranging from 0.1 to 0.9. After removing all ratings assigned by workers to itemsets for which they had expertise lower than the threshold, we split the dataset into training and test sets using a temporal cutoff as in the previous experiment (70% training set, 30% test set). We computed the RMSE for the same user-based collaborative filtering algorithm used previously (with a fixed-size neighborhood of 10 workers with the highest cosine similarity).

As can be seen in Figure 7.2, the RMSE decreases as the threshold  $\tau_E$  increases, confirming that the ratings for which workers are more expert are of better quality and enable the recommendation algorithm to produce more precise predictions. The RMSE immediately falls to  $1.819$  with  $\tau_E = 0.1$ , and reaches the lowest value of  $1.029$  with  $\tau_E = 0.7$ . As the value of  $\tau_E$  increases, we end up with too few remaining ratings and for  $\tau_E = 0.9$  the recommendation algorithm is not able to compute any prediction. As can be seen, the final RMSE is always lower than the one we obtained before removing lazy workers ( $2.202$ ).

**Effect of Weighting Ratings by Worker Expertise.** Filtering out the ratings of non-experts or flagged cheaters is a huge expense. In fact, they can still be of some value when aggregated. Another possibility is to weight the ratings by the expertise of the workers providing them when aggregating the items' ratings. To test the effect of this on the final aggregated ratings, we created two lists of aggregated ratings. In the first list, which we refer to the *unweighted* list, the ratings for each item were aggregated by taking the average over all the ratings provided for this item by all workers including lazy workers. In the second list, which we refer to as the *weighted* list, the ratings of each item were aggregated by taking a weighted average over all the ratings provided for this item by all workers (including lazy workers) such that each rating is weighted by the expertise of the worker that provided the rating at the time the rating was provided. To compare these two lists of aggregated ratings, we created a third list of aggregated ratings and used this list as a reference list. In this third list of aggregated ratings, the rating of each item was computed as the average of all ratings provided for the item by only “trusted” workers (excluding lazy workers).

We computed the RMSE (root mean squared error) for each of the two lists, the unweighted list and the weighted one, using the reference

list as the “true” average ratings. We obtained an RMSE of  $0.201$  for the unweighted list and an RMSE of  $0.077$  for the weighted list, with a reduction of  $62\%$  in average rating prediction. This clearly highlights the merits of weighting ratings by worker expertise when aggregating ratings. This can also be seen as another strategy for dealing with cheaters, instead of using a verification test and banning workers that do not pass it.

### 7.4.2 Parameter tuning

The goal of this set of experiments is to study the effect of the various parameters of our framework on cheater identification accuracy. To do so, we proceeded as follows: we generated a synthetic dataset and computed the accuracy of cheater identification for different values of the worker clustering threshold, the weights used in the utility function and the minimum skill threshold. We then tested the selected values on other larger datasets and a real-world one. Here we focused only on lazy workers as cheaters.

The synthetic dataset consisted of 100 fake restaurants, randomly placed in a 20-km diameter, divided into four non-overlapping itemsets  $I_1$ ,  $I_2$ ,  $I_3$ , and  $I_4$ , of the same size (i.e., 25 restaurants each). We generated 100 workers divided in the following way: 15 initial experts, 15 lazy workers and 70 trusted workers (i.e. providing truthful ratings). Each worker rated 40 restaurants, for a total of 4000 ratings with a value greater than zero (i.e., no don’t know or 0 ratings). The 15 initial experts in our dataset were divided into three non-overlapping groups each consisting of five experts. The first group liked itemsets  $I_1$ , and  $I_2$ , the second group liked itemsets  $I_3$  and  $I_4$ , and the third group liked  $I_1$  and  $I_4$ . In order to make rating generation simpler, we assumed all expert workers either liked or disliked all the items they know in any given itemset for which they provided ratings. This is a simplification of a real-world scenario where it is more likely that workers will like some items in an itemset and dislike others in the same

itemset. Similarly, our 70 trusted workers were divided into seven non-overlapping groups each consisting of 10 workers. The first three groups were similar to the three groups of experts, that is, they liked the same itemsets as the three groups of experts. The fourth group of trusted workers liked  $I_1$  and  $I_3$ , the fifth group liked  $I_2$  and  $I_4$ , the sixth group liked  $I_1$  only, and the seventh and final group liked  $I_3$  only. Rating generation for trusted workers was done as follows. All trusted workers gave a rating of 5 to items within the itemsets they liked with a 70% probability, and 3 (i.e., neutral) with a probability of 30%. The same happened for items they didn't like where a rating of 1 was generated with a 70% probability and a rating of 3 was generated with a 30% probability. Finally, the remaining 15 workers in our dataset were designed to be lazy workers with random ratings.

Using the above synthetic dataset, we tuned the different parameters in our framework. The first parameter is the similarity threshold for our clustering algorithm. Recall that our framework continuously re-clusters workers in the system as new ratings arrive. Our incremental hierarchical clustering algorithm merges clusters continuously until no two clusters can be merged (i.e., the closeness between any pair of clusters is lower than a threshold  $\tau_C$ ). We ran our algorithm with different values of  $\tau_C$  keeping all other parameters fixed to some randomly selected values:  $\beta_1, \beta_2, \beta_3$  and  $\beta_4 = 0.5$  for the utility functions and  $\tau_S = 0.3$  for the minimum skill threshold. We then computed the precision, recall and F2 measure for detecting cheaters, where

$$precision = \frac{\#true\_positives}{\#true\_positives + \#false\_positives}$$

$$recall = \frac{\#true\_positives}{\#true\_positives + \#false\_negatives}$$

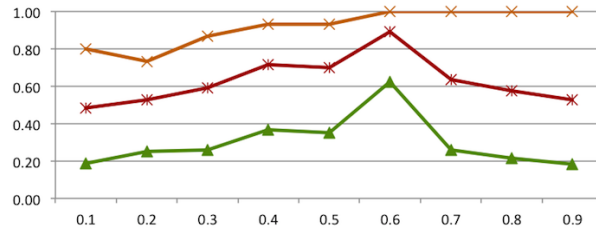
$$F2\_measure = (1 + 2^2) * \left( \frac{precision * recall}{(2^2 * precision) + recall} \right)$$

We used the F2 measure since we wanted to sacrifice a bit of precision in favor of a higher recall. That is, we want to detect as many lazy workers as possible, even with the price of falsely flagging some trustful workers as cheaters. This is not a problem in practice, since trustful workers will eventually pass the verification test after being flagged as cheaters.

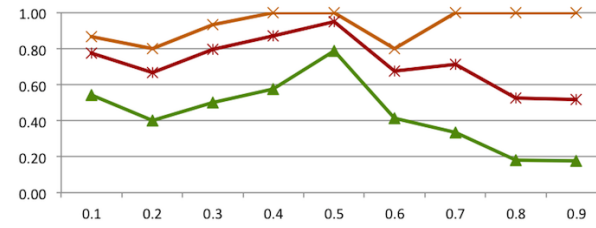
We obtained the highest F2 measure with  $\tau_C = 0.6$  (see Figure 7.3a). We also measured the quality of the clusters obtained with our algorithm with respect to the ideal set of clusters computed once all the ratings were generated. On average, 76% of workers were clustered correctly, with an 83% precision when the selected similarity threshold was used. In a similar fashion, we identified the best values for the other parameters in our framework:  $\tau_S = 0.5$  for the minimum skill threshold;  $\beta_1 = 0.3$  and  $\beta_2 = 0.7$  for itemset utility; and  $\beta_3 = 0.6$  and  $\beta_4 = 0.4$  for item utility. The results of this evaluation are shown in Figure 7.3.

To test the identification of cheaters on a larger dataset, we built four other synthetic datasets of bigger size, with 300 items each divided into six itemsets  $I_1$  through  $I_6$ , and 1000 workers. The set of experts was composed of 100 workers equally divided into five groups, same for all datasets. The groups have tastes like the ones presented for the smaller dataset, but we added also an itemset-independent taste: one group liked only items whose id was a multiple of three. This last group represents a more realistic group of workers who like and dislike items within the same itemset and across itemsets. Experts' ratings were generated with the correct rating (i.e., 1 for items the expert didn't like and 5 for items she liked) with 80% probability and the neutral rating (with value 3) with 20% probability.

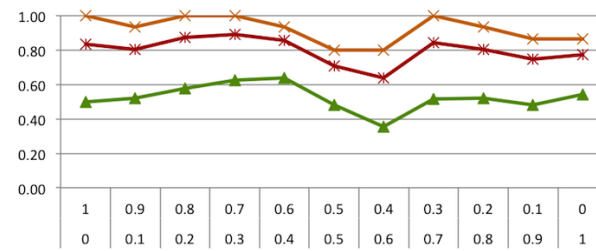
The main difference between the four datasets in this evaluation is the number of lazy workers. Dataset A had 100 cheaters, dataset B had 200 cheaters, dataset C had 300 cheaters and dataset D had 400 cheaters. Trusted workers were equally divided into 10 different groups, five of which



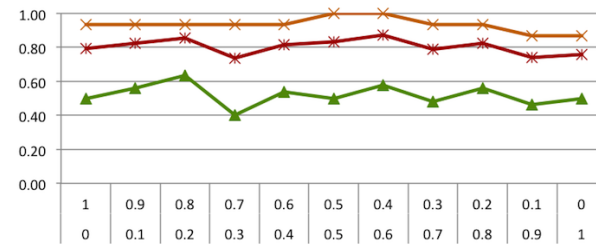
(a) Best value for  $\tau_C$ .



(b) Best value for  $\tau_S$ .



(c) Best values for  $\beta_1$  and  $\beta_2$ , with  $\beta_2$  values on top and  $\beta_1$  values on bottom.



(d) Best values for  $\beta_3$  and  $\beta_4$ , with  $\beta_4$  values on top and  $\beta_3$  values on bottom.

▲ Precision 
 ✕ Recall 
 ✱ F2 measure

(e) Legend.

Figure 7.3: The best values for system parameters.



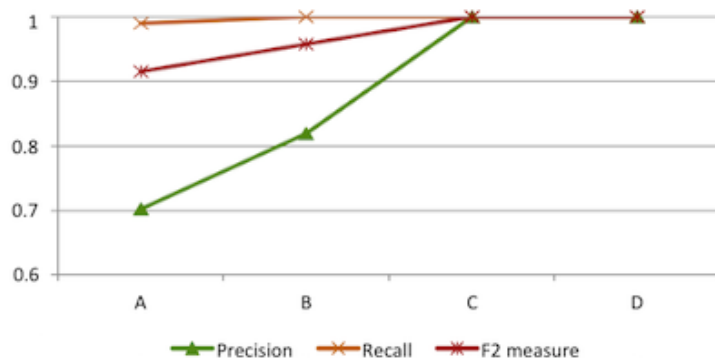


Figure 7.4: Results of the 4 synthetic datasets with 1000 workers.

correspond to the ones of the experts while the rest are composed in the same way, but with different combinations of itemsets. Trusted workers gave the correct ratings (either 1 or 5) with a 70% probability and they gave the neutral rating (i.e., 3) with a 30% probability. Each dataset contained around 120,000 ratings.

As shown in Figure 7.4, our framework performs consistently well for all four datasets, reaching full precision and recall for dataset D (i.e., as the number of cheaters increase). In addition, each task was run in about 2 seconds on average, from the received rating to the reply with the next item to rate. This result clearly indicates the feasibility of our approach as a web service for crowdsourcing rating tasks.

Finally, we ran our framework with the best parameter values determined by the previous experiment on our real dataset. We obtained a precision of  $0.727$ , a recall of  $0.800$  and an F2 measure of  $0.784$ . These results confirm the ability of our framework to correctly identify cheaters in a real setting.

### 7.4.3 Utility function experiments

Our utility function has no influence on the identification of cheaters: cheaters will sooner or later reveal themselves as they rate more items,

regardless of the order in which items are presented. The importance of the utility function is the time needed to identify cheaters. Clearly, the sooner they are identified, the better. If the framework needs to collect many ratings before identifying cheaters, this would be costly and more importantly, it could happen that a cheater might stop giving ratings before the framework had had the chance to identify her as one. In this case, the ratings provided by this unidentified cheater would be considered reliable.

To test the effect of our utility function on the overall performance of the system, we compare it to two other baseline utility functions: i) a recommendation-based utility function, in which the next item shown to the worker is the one recommended to the worker according to the ratings she already gave using an off-the-shelf recommendation system, and ii) a random utility function, in which the next item is randomly selected. Recall that our proposed utility function is based on the number of ratings already assigned to an itemset, the profile of the worker for the itemset, the number of ratings already given for an item and the similarity of the item with other items rated by the worker. To test which utility function performs best, we analyzed the number of ratings the framework asked each cheater before identifying her, using our real dataset of restaurants. Using our real dataset, on average the random utility function needed to show 25 items and the recommender-based utility function needed to show 32 items, while our utility function needed to present only 17 items. These different results are statistically significant according to t-tests between the pairs (p-values: 0.0001, 0.008, 0.03,  $\alpha$ -level: 0.05). This means that our utility function identified cheaters at least 32% earlier than the other two functions.

Another important aspect of our utility function is that it keeps the number of ratings balanced over items which is a main goal of our data

Table 7.1: Standard deviation of number of ratings per item.

$N$	Our utility	Recommendation-based utility	Random utility
150	0.2	0.53	0.59
300	0.99	0.53	0.57
600	1.2	0.67	0.57
1200	1.32	1.67	1.11
2000	1.94	1.81	1.75

acquisition that distinguishes it from a recommendation system. When the framework chooses which item to show next, it gives higher priority to items that have fewer ratings, balancing in this way the number of ratings across items. To verify this, we analyzed how many ratings items had after  $N$  data acquisition rounds, for different values of  $N$ . We used standard deviation to compute rating distributions. The results are shown in Table 7.1. We can see that all utility functions have close values of standard deviation, with our utility function having smaller values for the first 150 ratings. A smaller deviation means that the number of ratings across items is quite balanced. When we consider a higher number of ratings, the standard deviation increases even with our utility function. This is mainly an effect of the initial items proposed to the worker when she arrives. Since the system aims to gather enough ratings per worker to be able to cluster them, the first items proposed to workers are those with the highest number of ratings. We thus end up with a small set of items that have more ratings than others, causing this problem of unbalanced ratings.

#### 7.4.4 Malign workers experiments

So far, we have only considered lazy workers as cheaters, i.e. those workers who provide random ratings. There are other ways for cheating and a par-

ticularly appealing category of them are the malign ones. Malign workers are workers who intentionally give misleading ratings to particular items to reduce or raise their average ratings.

In this set of experiments, we added malign workers to our real and synthetic datasets and tested how many of them are correctly marked as cheaters. For the synthetic dataset, these malign workers provided “truthful” ratings for a percentage of the items by following the same behavior of some of the experts in the system (as these behaviors are well defined). For the real dataset, the malign worker followed the behavior of the majority of the other workers (i.e., giving a positive rating when the majority gave a positive rating and vice versa). For the rest of the items, the malign workers provided opposite ratings to those provided by the experts or the majority depending on the dataset, reducing or raising the average ratings of these items.

We start by testing how many misleading ratings these workers had to give before being identified as cheaters. In the synthetic dataset used for parameter tuning (with 100 workers), we added 24 malign workers divided into four groups of six workers with different percentages of misleading ratings: 10%, 20%, 30% and 40%. The framework identified 87% of the malign workers on average (21 of 24), and 71% of the ones that it missed to identify had only 10% of misleading ratings. Considering only the workers with a higher percentage of misleading ratings, the framework identified on average 95% of the malign workers. We conclude then that the framework is able to correctly identify almost all malign workers when they give at least 20% of misleading ratings.

We also computed the recall of malign workers’ identification as we vary the clustering threshold ( $\tau_C$ ). Since workers are marked as cheaters when they fail to join clusters, a different value for this parameter could increase or decrease the amount of misleading ratings the workers should provide

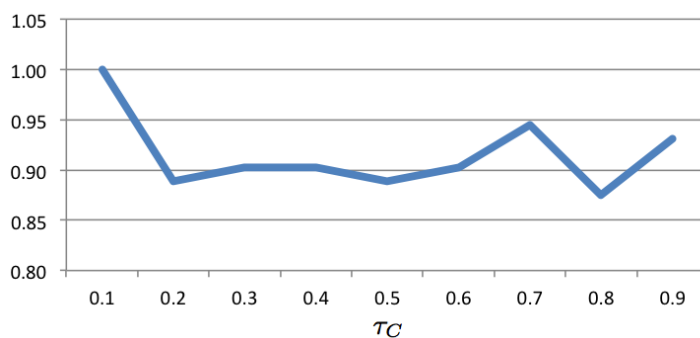


Figure 7.5: Average recall of malign workers identification.

to be identified as cheaters by our framework. Figure 7.5 shows that the recall remains stable with varying  $\tau_C$  values. On average, 75% of malign workers that were not identified as cheaters had only 10% of misleading ratings. This confirms the previous limit of 20% of misleading ratings as the minimum amount of misleading ratings that a malign worker has to provide to be identified as a cheater by our system.

Finally, we confirmed the above results using our real dataset. We added 10 malign workers with 20% of misleading ratings and the rest of the ratings following the majority of the other workers, and the framework correctly identified 93% of the malign workers on average. These results are quite promising, but marking malign workers as cheaters is only half of the work. To complete the work, malign workers should not be able to pass the verification test. However, since malign workers are aware of their misleading ratings, they will be able to reproduce their ratings when the verification test is run. We leave the identification of a different verification test that is hard to pass for malign workers but not for trustful workers falsely flagged as cheaters to future work.

## 7.5 Conclusion

We presented a crowdsourcing platform to acquire reliable ratings of items. Our data acquisition platform differs from existing crowdsourcing systems and recommendation systems because it targets the most expert users to provide ratings for items with the fewest number of ratings. Our system relies on incremental clustering to identify cheaters and a carefully-designed utility function to assign items to rate to the most expert workers. Our experimental evaluation on both synthetic and real restaurant datasets showed that detecting cheaters, acquiring ratings from expert workers only, and automating the rating acquisition process all have a positive impact on both the cost of acquiring reliable ratings and on improving recommendation accuracy in popular recommendation systems.

With a crowdsourcing platform implementing the presented framework, we could easily and quickly collect the initial ratings needed to start a recommender service for leisure activities in a new area, collecting high-quality ratings for the new items added into the system. With this platform, we can easily identify whether a worker is a local or not for the area we are considering and acquire ratings only if the worker demonstrates to be trustworthy.

# Chapter 8

## Planfree - a Restaurant Recommender Service

### 8.1 Introduction

We studied how to collect people's feedback and how to use it to build high-quality personalized recommendations. In this chapter we present Planfree, the prototype of a restaurant recommender service the which we applied the results obtained in our studies. The prototype has been developed as a tool to be integrated in Toolisse ([www.toolisse.com](http://www.toolisse.com)), a platform of travel solutions.

The fact that recommendations here are limited to restaurants, i.e. places for having lunch or dinner, does not mean that the same service cannot support recommendations for other leisure activities. This solution is generic for leisure recommendations, but is limited to restaurants only to keep the system smaller and to make its characteristics more visible and clear.









Osteria Vineria San Martino			
Via San Martino 42, Trento			
 with visitors	How was your experience? 	 with friends	How was your experience? 
 romantic	How was your experience? 	 lunch break	How was your experience? 

Figure 8.1: Interface for rating collection.

## 8.2 Satisfying the Requirements of Recommender Systems

In Section 1.1 we introduced the requirements that recommender systems have to satisfy and in the previous chapters we have studied the solutions that can be applied. Here we explain which solutions we adopted for our restaurant recommender service.

### 8.2.1 Learning user's tastes

One important decision to take is about how to collect people's feedback. As we have seen in Chapter 6, the favorite rating scale on mobile devices is *5-star*. Since it was already identified as the best rating scale also for desktop computers, we can use this rating scale without the risk to have dissatisfied users.

We have seen in Chapters 4 and 5 the importance of *purpose-based ratings* to collect contextual information together with the ratings and to build more tailored recommendations. For this reason we ask 4 ratings for each restaurant: with visitors, romantic, with friends and lunch break. Each



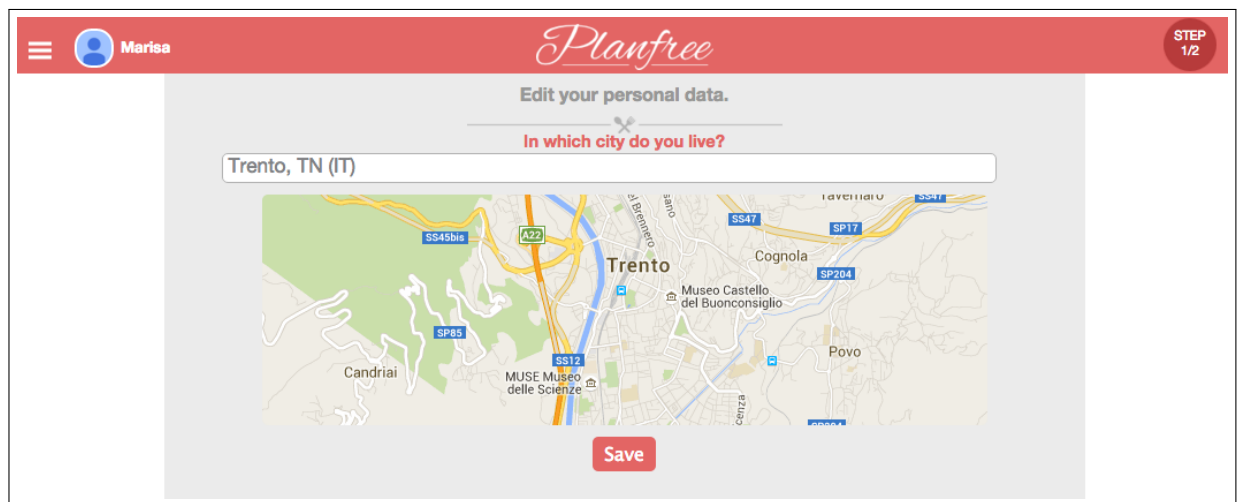


Figure 8.2: First-time user: inserting home town.

purpose is easily distinguished by the meaningful icon and the different color that represent it (Figure 8.1).

When a new user arrives, we have to learn her tastes before she requests recommendations, otherwise we can present only generic recommendations. We decided to build a specific path for *first-time users*. People enter the service after the mandatory login, implemented through two major social logins: Facebook and Google. While returning users access directly the recommendation list with their registered settings, first-time users are first forced through a two-steps profile-filling procedure. First, the new users are asked to insert their home town or the city in which they feel local the most (Figure 8.2). Then, if the service is already available in their city, they are asked to rate the restaurants they know there (Figure 8.3). In this way, we can start knowing some preferences of these new users before they request any recommendation.

By asking people their home town or the city in which they feel local the most, we are able to recognize whether they are *locals or tourists* for the city they are asking recommendations for. In Chapter 5 we presented the importance of locals' opinions, as they are more knowledgeable about the

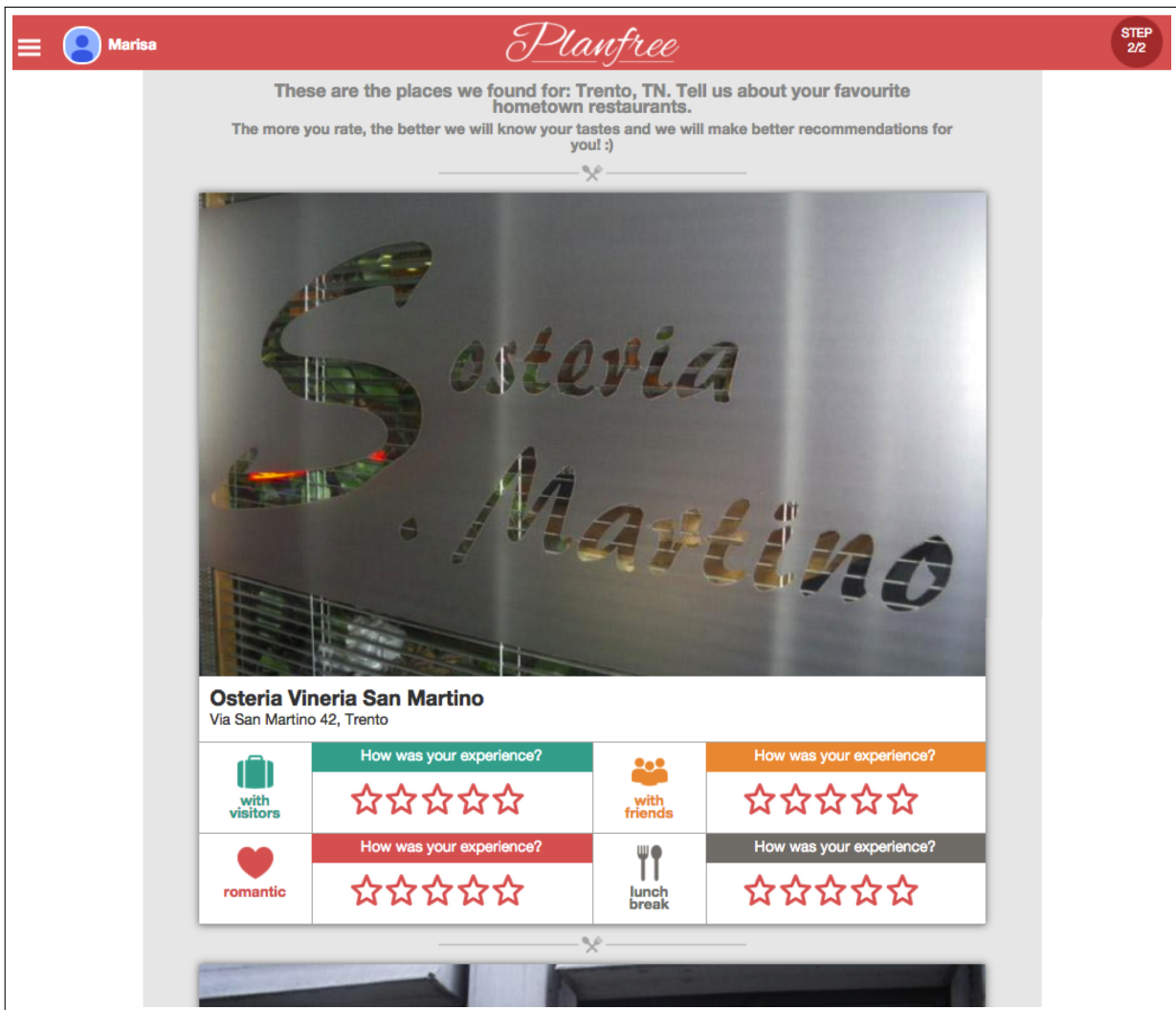


Figure 8.3: First-time user: inserting ratings for home town restaurants.

offerings available in their city, so knowing this information let us consider locals' ratings with the adequate importance.

Since the profile procedure does not force people to share their feedback about the known restaurants, i.e. people can easily skip the task if they don't want to complete it for any reason, we still have to be able to build their recommendation lists when they request them. For this reason, we integrate the personalized recommender algorithm with a generic one, based on the average of ratings.

### 8.2.2 Making recommendations

In Chapter 6 we analyzed the collaborative filtering recommender algorithms, searching the one that builds the best recommendations, both according to objective metrics (i.e *precision*, *recall* and  $F_{0.5}$  – *measure*) and to user evaluation. We found that user-based collaborative filtering is the more precise and satisfactory, in particular when combined with 5-star ratings. Unfortunately, *user-based collaborative filtering is not scalable*, and when the number of users, items and ratings increases, it takes too much time to compute the requester’s neighborhood and compute the recommendations. We want to provide a fast and reliable service to mobile users too: they usually want to get the requested information immediately, waiting maximum few seconds. This means that we cannot make them wait for a couple of minutes every time they access our service.

To get a recommender service fast and scalable, we moved our attention to *cluster-based recommender services*. They have the advantage of computing clusters of users in advance, and even the predicted ratings for each item and each purpose for the cluster can be stored in advance. When a request of recommendation arrives, the predicted ratings are already available and only context-based considerations have to be done. This let us quickly answer when a request arrives, but the clusters need to be periodically recomputed as new ratings arrive.

In Chapter 6 we analyzed the cluster-based clustering filtering with two different clustering algorithms: hierarchical clustering and k-means. Despite hierarchical clustering gives very good results, very close to the ones of user-based collaborative filtering, it is not scalable and even with only one thousand users it takes too much time to recompute clusters, making them useless as new data arrive while this computation takes place. For this reason, we temporarily adopted k-means as clustering algorithm.

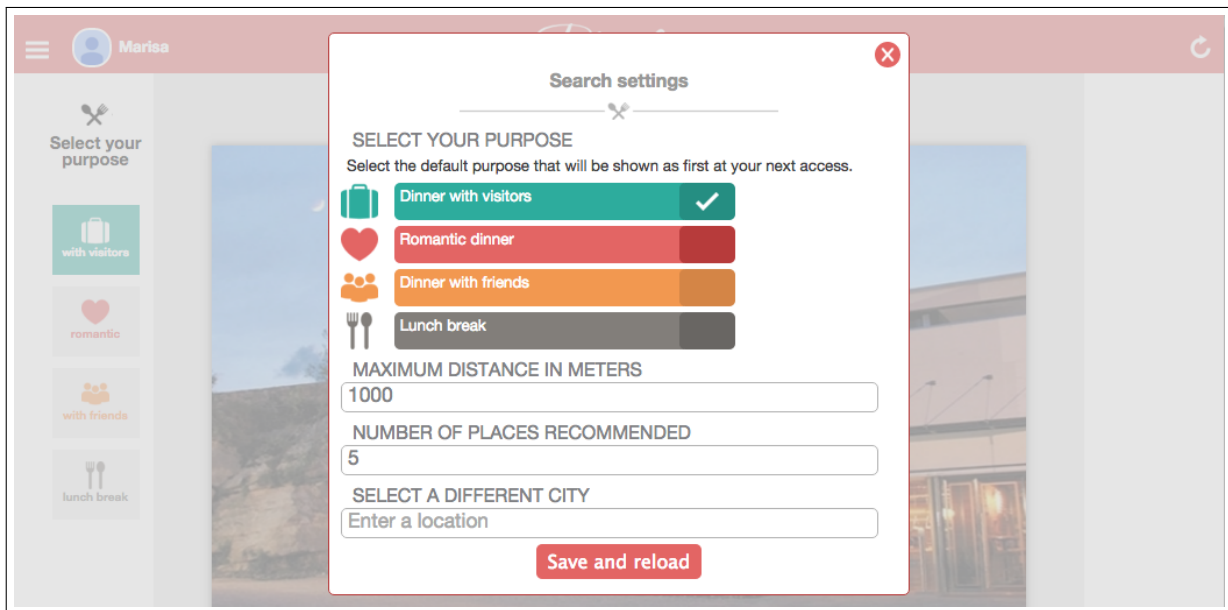


Figure 8.4: Recommendation settings.

The clusters it computes can be of low quality, but they are computed quickly and we can provide some recommendations to our users. We will search for better algorithms for the next versions of this service.

Once clusters and cluster-generic recommendations have been computed and stored, the recommender algorithm needs to tailor the results to the requester’s needs. First of all, user’s location is considered, filtering out the places that are too far from her. The user can indicate to the algorithm what “too far” means to her according to her current situation by adjusting the settings, shown in Figure 8.4: she can indicate the maximum distance in meters she is willing to walk/drive to reach the best place for her needs.

The settings let users specify also the number of recommendations they are willing to receive and the city they are interested in. Thanks to this, the service is flexible and people can also use it to plan in advance their lunches and dinners for a future visit to a different city.

Then, the algorithm has to consider the purpose indicated by the requester. Despite clusters are computed considering the full set of ratings

collected till their computation, the predictions for each item are computed taking purpose in consideration. For example, the prediction for item  $i$  and purpose  $p$  is computed as average of the cluster's ratings for that item and that purpose, without considering what the users within the cluster thinks about the item for other purposes. The recommendation list is computed by sorting the items according the prediction computed for the requested purpose.

Finally, the previous ratings of the requester are considered to personalize the recommendation list: the places the user already rated negatively are removed from the list. Now we have the complete recommendation list for the requester, satisfying all her needs. The service returns the top- $k$  places, with  $k$  being the number of items the user requested (by setting it in the settings).

To make the recommendation list consumption easier from mobile devices, we suggest the following settings, which are the default values: 5 items in the recommendation list, which contains only places within 1000 meters from user's current position. A distance of 1000 meters could seem big, but it can easily be covered in 20 minutes by walking. Such distance makes the user sure that the top restaurant the service recommends is the best one she can reach from her position by foot.

More details about the algorithm can be found in Section 8.3.2.

### **8.2.3 Availability everywhere and anytime**

We have seen that one of the major challenges is the availability of the service everywhere and anytime. This suggests to build a service that can be easily accessed by any device: desktops, tablets and smartphones.

To solve this problem, we built Planfree platform as a Web application that can be accessed by regular web users through any standards-compliant, desktop/mobile Web browser. The Web interface is responsive:

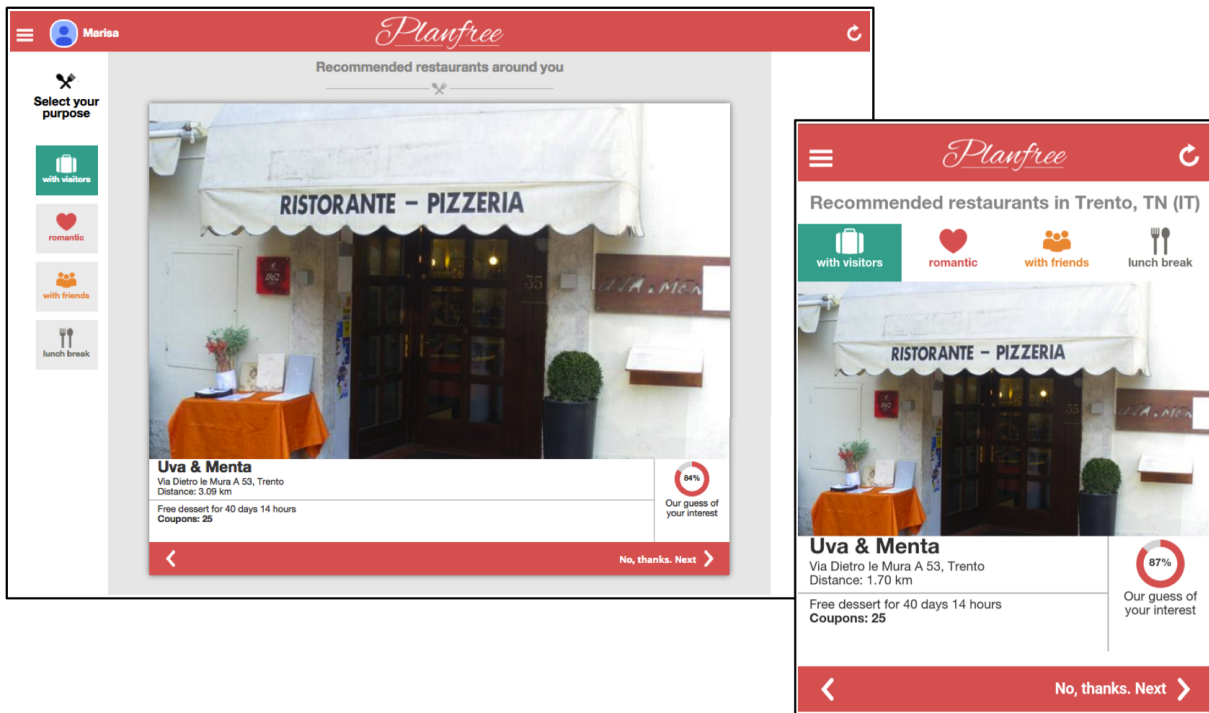


Figure 8.5: Recommendation page for mobile (right) and for desktop (left).

it adapts to the device size. Figure 8.5 shows how the recommendation page is shown on mobile and desktop. Moreover, the platform provides an application programming interface (API) for developers that can be accessed through third-party clients. Through this solution, the service can be accessed through the Internet without any need of installing any application, but at the same time the platform is ready for supporting device-specific applications that can access the Planfree backend through the Internet. More details about the platform are available in Section 8.3.

To make the recommendation list easy to consume and to point user attention to the top positions, we show only one recommended restaurant at a time, starting from the first position and going down through the list. In this way, people accessing through the smartphone can immediately see all the details about the restaurant that can affect their choice: the restaurant name and image, how far it is and how much we recommend it (as an

indicator of the rating we predict the user will share after experiencing it). With all these details, the user can decide whether to go to this restaurant or to swipe to the next recommendations without the need to access the full details of the restaurant, i.e. without moving to a different page of the website.

### 8.2.4 Getting interest of users and restaurateurs

With Planfree we are presenting a novel recommender service, but from the outside it looks like all the others. Our main strength is the quality of recommendations, based on the results of our studies. This characteristic is, however, not visible by people until they try the service and compare its results with the results of other competitors (TripAdvisor, Foursquare, The Fork ...). This means that we need to add some features to attract people and make them try our service (and discover how good it is).

To solve this problem we decided to offer a couponing service integrated with the recommender one. Restaurateurs are invited to enter our platform and request the right to control their restaurant's page. Through it, they can directly provide the full information about their restaurant and offer some discounts in form of coupons. Involving restaurateurs in the platform, we have also the advantage of providing curated information about the places, improving the quality of the service.

These discounts are advertised to people if the restaurant is available in their recommendation list, as can be seen in Figure 8.5: this means that only people that would like the experience at that restaurant, according to our recommender service, will have the chance to get the coupon. The discount attracts new people to the restaurant and, since the restaurant was recommended to them, there is a good chance that these people will like the experience and return in the future. In this way, restaurateurs

## CHAPTER 8. PLANFREE - A RESTAURANT RECOMMENDER SERVICE

☰ Marisa *Planfree*


### Uva & Menta

Via Dietro le Mura A 53, Trento

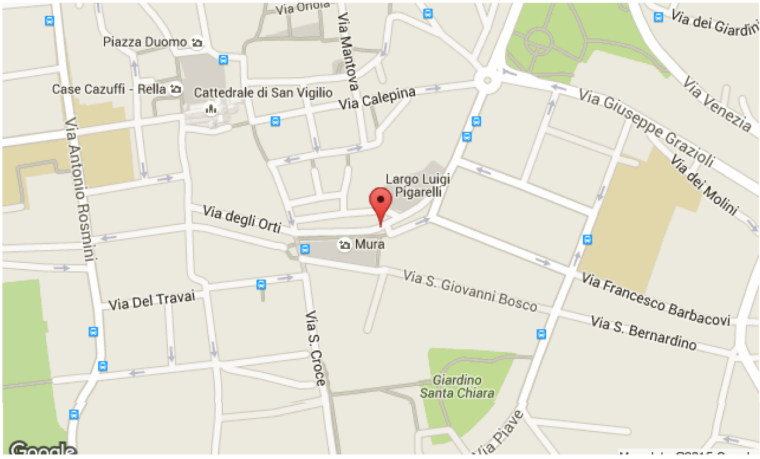
< Back

**Free dessert**  
Come here for your lunch and you will get a dessert for free  
25 coupons, available for 40 days 14 hours

Get coupon



Restaurant and Pizzeria also gluten-free pizzas. Large selection of beers.



📍 Via Dietro le Mura A 53, Trento (TN, IT) ☎ 0461 190 3162

Figure 8.6: Restaurant page with discount details.



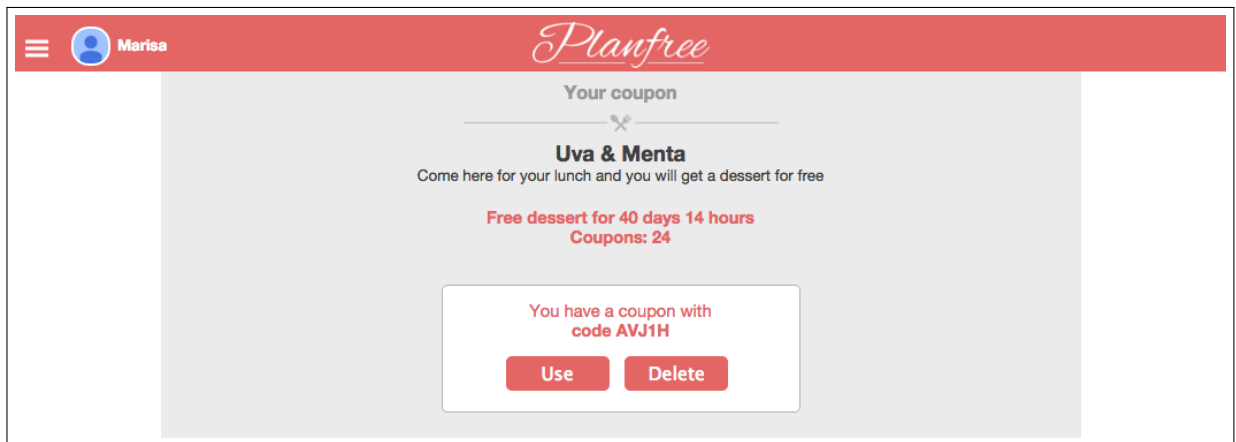


Figure 8.7: Details about the obtained coupon.

can get high-quality advertisement and high return of investment from the small expense of providing few coupons.

Advertising the availability of coupons through Planfree, people are motivated to enter the service, provide the initial ratings and try the recommendations. Once first-time users discover the quality of our recommendations, they hopefully will become returning users and will recommend Planfree to their friends and relatives.

The coupons can be requested from the restaurant page (Figure 8.6), and the obtained coupon has the details shown in Figure 8.7. The personal list of acquired and used coupons can be accessed from the menu, keeping track of all the offers the user is willing to take advantage of.

We did not have the chance to test the effect of coupons and how much they will attract restaurateurs and users, but this model have been already used in different contexts. For example, The Fork is a restaurant booking service and uses discounts as advertisement and as a motivation to prefer their service instead of using one of its competitors.

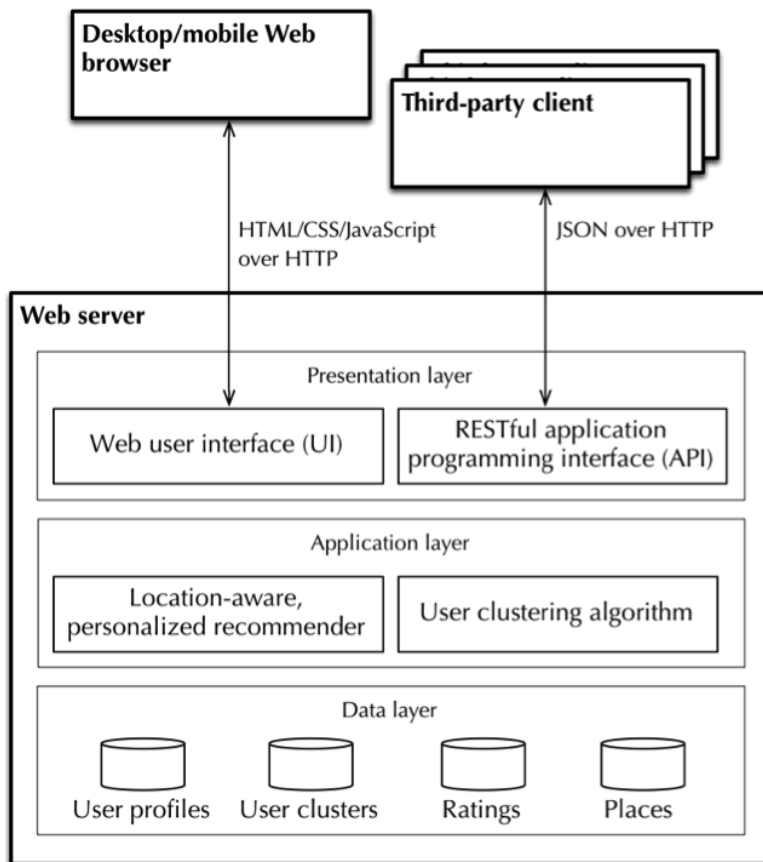


Figure 8.8: Conceptual architecture of Planfree.

### 8.3 Architecture and Implementation

Planfree comes as both a Web application for regular Web users and an application programming interface (API) for developers. The former can be accessed via any standards-compliant, desktop/mobile Web browser, the latter through third-party clients.

The architecture follows the conventional client-server pattern of Web applications and can be split into three layers (see Figure 8.8):

- The presentation layer provides the access to Planfree and its features. It comprises the Web user interface for human consumption

and the RESTful application programming interface for consumption by software agents.

- The application layer hosts the two core ingredients of the Planfree approach to recommending nearby places (e.g., restaurants or bars): the location-aware, personalized recommender algorithm and the user clustering algorithm. The former computes place recommendations for users on the fly, the latter clusters users into groups with similar interests/profiles both online (for incremental updates of the clusters) and offline (to periodically re-compute the whole set of clusters).
- The data layer hosts the data the two algorithms work on: user profiles, user clusters, ratings, and places. User profiles are automatically created upon the registration of new users and can be edited by the users. The ratings are provided through an own form part of the UI. The user clusters are computed by the clustering algorithm. The places are pre-loaded (e.g., taken from OpenStreetMap or the Toolisse database).

The whole backend runs in the cloud (Google App Engine), is written in Python, HTML, CSS and JavaScript, and uses Google Datastore for data management.

### 8.3.1 Data layer

The diagram in Figure 8.9 represents the conceptual data model underlying Planfree.

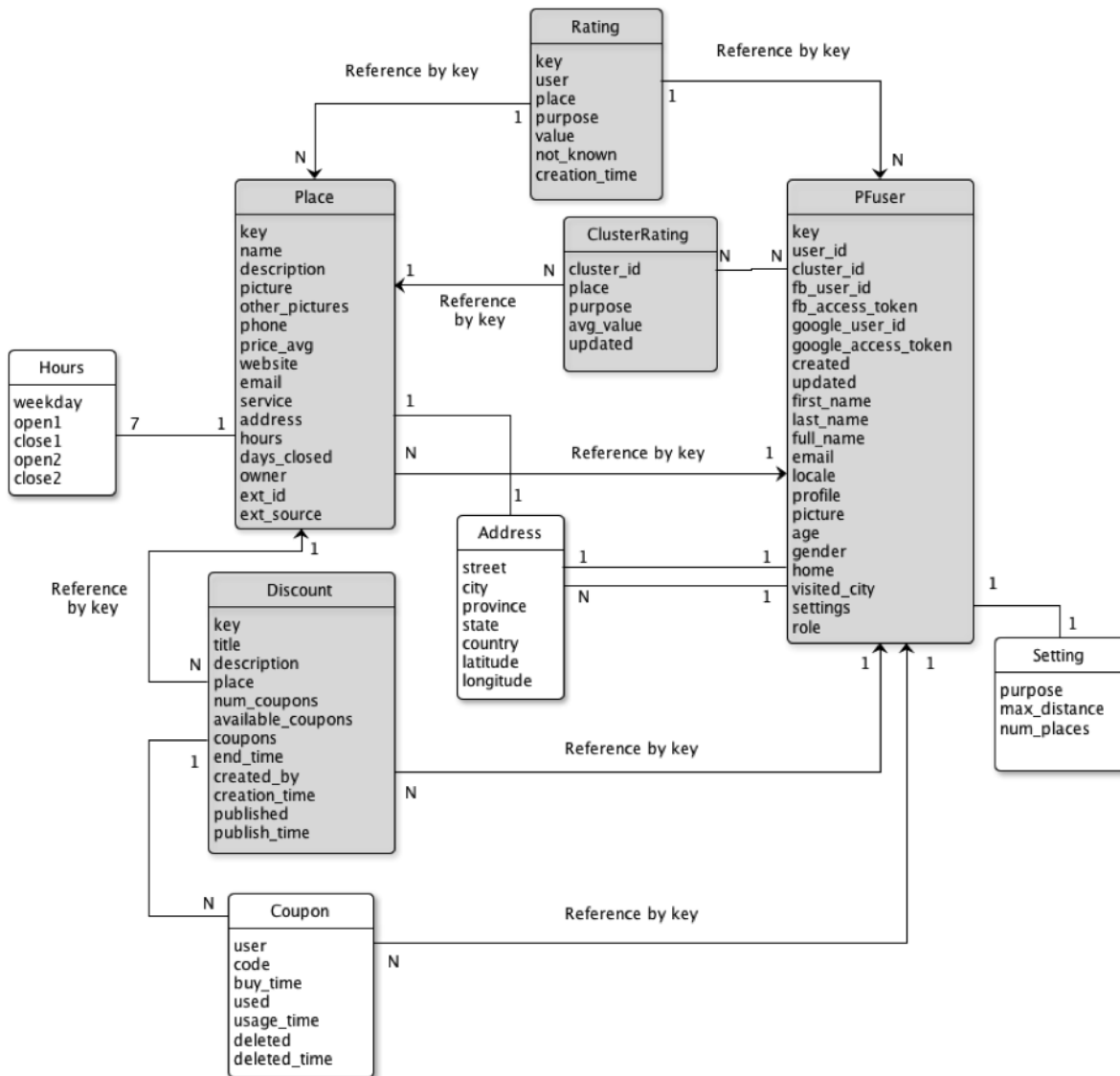


Figure 8.9: Conceptual data model of Planfree's data layer.

### 8.3.2 Application layer

This layer is responsible for elaborating the data and providing the needed information to the user. The most important task here is executed by the recommender algorithm.

We use a cluster-based recommender algorithm: it groups users in clusters using k-means clustering algorithm, and the opinions of the cluster are

**Algorithm 2** Compute clusters

---

**Input:** all users and their ratings, the number of clusters  $k$ ,  $t_C$  threshold**Output:** centers of clusters,  $cluster\_id$  set in each user

```
while true do
  for each user  $u$  do
    for each center  $c$  do
      compute user  $similarity(u, c)$ 
    end for
     $u.cluster\_id = c.cluster\_id$ , for  $c$  with max similarity
  end for
  for each cluster do
     $new\_center = avg\_ratings(usersincluster)$ 
  end for
  if  $min(similarity(old\_ca, new\_ca)) < t_C$  then
    return
  end if
end while
```

---

used to build recommendations for the requester. K-means algorithm has been selected because it can be implemented using MapReduce technology and in this way it scales well.

**CLUSTERING:**

**Initial step** The users that already rated some restaurants are grouped into clusters.

**Incremental step** When a new user arrives, she provides some initial ratings and, based on them, she is added to the closest cluster.

**Re-computation step** Once in a while (once a week/month), clusters need to be recomputed, to take into consideration also the ratings added or updated since the last cluster computation. This operation can be planned to execute when the server(s) have low workload and can concentrate the resources in this task.

---

**Algorithm 3** User similarity (based on euclidean distance)

---

**Input:** pair of users  $x$  and  $y$  and their ratings**Output:** similarity value $n = \#$  of common ratings of  $x$  and  $y$  (i.e. ratings of same place for same purpose) $r(x, pl, pu)$  = value of rating of user  $x$  for place  $pl$  and purpose  $pu$  $sum = 0$ **for** each place  $pl$  and purpose  $pu$  in common ratings **do** $sum = sum + square(r(x, pl, pu) - r(y, pl, pu))$ **end for** $similarity = 1 / (1 + (sqrt(sum) / sqrt(n)))$ return similarity

---

## RECOMMENDATIONS:

The algorithm receives a user query containing the following information: purpose, number of recommendations needed and filtering information for places.

The filtering information for places at the moment are related to geolocation: the user position (latitude and longitude) and the maximum distance the user is willing to walk to reach the place.

The recommender first retrieves the list of places that are within max distance from user location, then it gets users cluster and build the predicted rating using users cluster ratings.

The algorithms 2, 3 and 4 show more details in pseudocode.

### 8.3.3 Presentation layer

The application exposes its services to both real users, through a web user interface, and other applications, through the REST APIs. The web user interface has been designed to be easy to use and responsive: it adapts to any device, both desktop and mobile.

**Algorithm 4** Recommend

---

**Input:** center of user cluster, user, user location (given by user latitude and user longitude), user maximum distance, purpose , number of recommendations to compute ( $n$ )

**Output:**  $n$  places

*placelist* = list of places within maximum distance from user location

**for** place  $p$  in *placelist* **do**

    score of  $p$  = ratings of cluster center

**end for**

return  $n$  places with higher score

---

### 8.3.4 Implementation and code base

The project is stored on github at the following address: <https://github.com/sphoebus/rockshell>.

The application is built to run on Google App Engine, the cloud service provided by Google. It uses a Datastore to store all the data, and the definition of the data model depends on this no-sql database.

The project is implemented in Python, using the libraries webapp2 and jinja2. Webapp2 is used to make the application able to receive and respond to HTTP calls. Jinja2 is a templating library, which allows us to easily fill data in the html/css/js pages that are returned by the user interface calls.

To allow geolocation-based queries for places, we use a small table in Google Cloud SQL, i.e. a MySQL database within the Google Cloud. This table contains only place keys and coordinates and here we can perform a query which retrieves all places within a rectangular area, while with datastore it is not possible.

To cluster users very quickly, we implemented k-means algorithm using MapReduce procedures. In this way, the computation for each user can be parallelized, getting the results very quickly.

For the user interface, we used HTML, CSS and JavaScript. We used JQuery and underscore libraries. The first one let us handle some minor http calls and update the visualized html page without the need to reload it entirely. Underscore library is used for building html templates to be used by JavaScript. These templates are used to handle place lists pagination in the browser, without the need of extra communication with the server for only visualization changes.

The responsiveness of HTML pages is obtained only through css, using percentage sizes and media queries, without the need of extra libraries.

## 8.4 Tests

To verify that we matched our goals of building a platform that can provide an awesome experience to thousands of users, we run some tests about scalability and usability.

### 8.4.1 Scalability test

The application has been developed keeping in consideration scalability. In particular, the recommender algorithm has been chosen according to its ability to scale well as the number of users grows. The algorithm we selected for cluster-based collaborative filtering is k-means and it can be implemented with MapReduce: at each step, a fixed number of “centroids” are selected and each user is associated to the most similar one (map step), then the new centroids for the next step are computed (reduce step). The centroids are some representative users, one for each cluster, which are computed by averaging the ratings of all users within the cluster. At the beginning some users are randomly selected as initial centroids. Moreover, we implemented Planfree in Google App Engine, which provides more resources and new instances when the traffic increases, splitting the work



over more machines automatically to keep performance high. The combination of the MapReduce algorithm and Google App Engine guarantee that Planfree scales well.

To prove the efficacy of the algorithm and the quick response of the application when a user requests her recommendations, we tested Planfree with a synthetic dataset generated randomly and containing 1000 restaurants, 1000 users and 10000 ratings.

Average time to get the restaurants around the user: 0.11 seconds

Average time for loading full recommendations (get places, get cluster ratings, get user ratings and, if needed, compute overall average ratings): 0.75 seconds

Average time for cluster computation (tested with 10 clusters): 2 hours

### 8.4.2 Usability test

The usability of the application has been tested online, and 11 participants completed the evaluation. The test consisted in performing 4 tasks that cover the main functionalities of the application: rate restaurants (task #1), browse through recommendations (task #2), get a coupon and spend it (task #3), and change settings (task #4).

In general the application resulted easy to use and pleasant in all tasks, but there is space for improvement in particular to support first-time users. For each task, very few errors (self-reported usage mistakes, not system errors or bugs, such as accidentally closing a dialog window without having filled the respective form) were made by participants, with the higher number of errors obtained for the task of rating restaurants (Figure 8.10): 7 participants made 1 to 3 errors, but only 2 of them found it difficult to recover from their errors and anyway did it on their own without side effects. These errors are mainly due to the navigation through the application. In

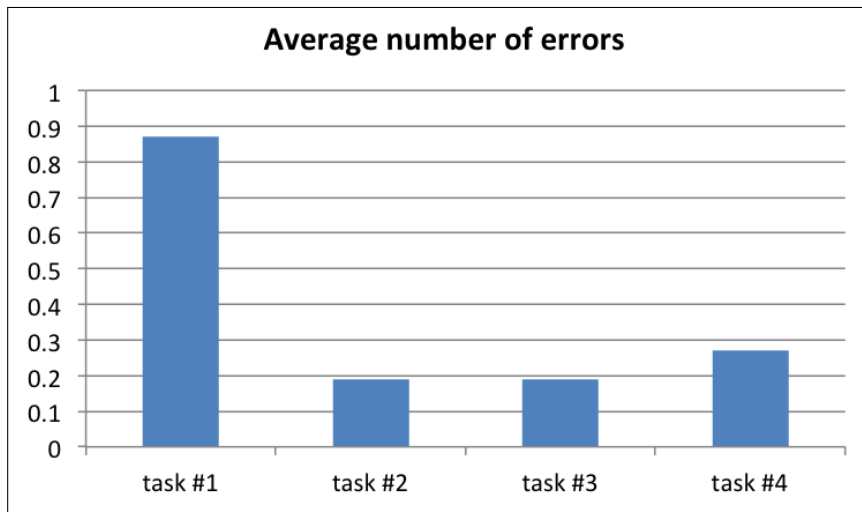


Figure 8.10: Average number of self-reported usage errors in usability tasks.

the rating page, at the bottom there are both arrows to navigate through the restaurant list (which is paged) and the button to end the rating task. These buttons turned out to be not visible and clear enough, and they have been improved after this evaluation.

A different error was identified for a particular device/browser combination. When you access the website using Firefox on an Android smartphone, there are issues with the autocomplete provided by Google Maps API: the suggestions are not visible until a space or comma is pressed and the first suggestion is almost impossible to select. We will search for solutions and workarounds. The few errors made for the other tasks were all easy to recover and none of them was severe.

The task that was considered the slowest was the one related to coupons: only three restaurants had available coupons and it was hard to find them if they were not between the first recommendations. This is not a negative result: we wanted users to see coupons only for restaurants we think they would like, and this is exactly what happened. The interfaces were all considered primarily pleasant (Figure 8.11), but the interfaces for the tasks “rate restaurants” and “get a coupon and use it” received also some

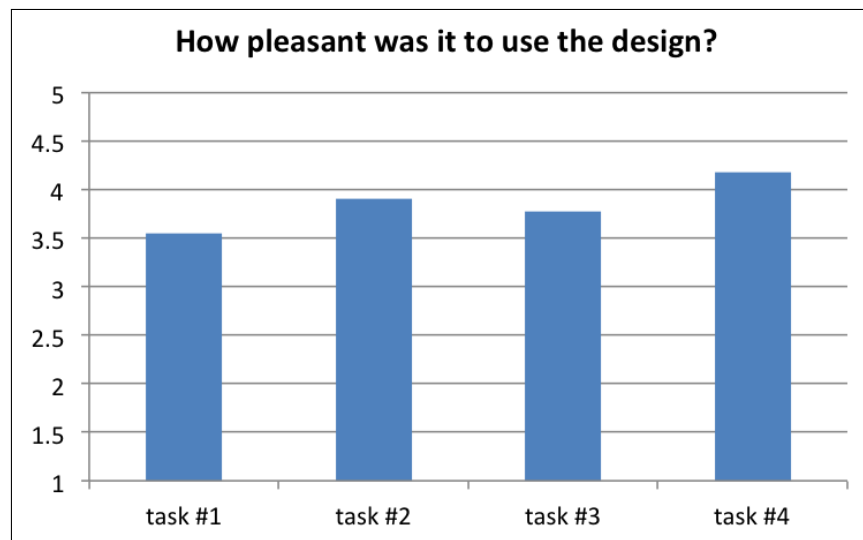


Figure 8.11: “How pleasant was it to use the design?”, in a scale from 1 (not pleasant at all) to 5 (very pleasant).

comments on the possibility of improvements to make them even easier to use.

As can be seen in Figure 8.12, participants expressed a higher likelihood to use Planfree to get recommendations and coupons for other cities they don't know yet, while in their home city they will be interested more in coupons than in recommendations. A couple of participants would not use this application in their home city since they don't see any benefits in choosing it against TripAdvisor and similar services.

Between the features people would like to see in the application, booking of a table is the most requested one, followed by more user-created content, like comments, pictures (not from the owner) and feedback in general.

## 8.5 Research Exploitation

From a research perspective, the project had important results, but more work is needed to improve the accuracy of predictions with low number of ratings, and specifically low number of ratings per place per purpose. This

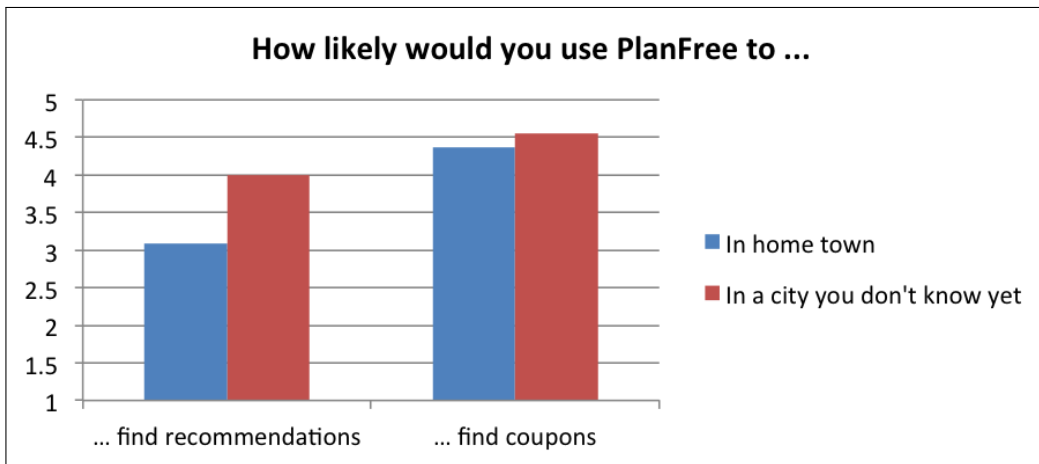


Figure 8.12: : How likely participants would use Planfree, in a scale from 1 (very unlikely) to 5 (very likely).

is a common scenario even in the most widely used recommender systems such as Yelp and TripAdvisor, and it is also a common problem with new places (or, even more interestingly, in old places with new management). Understanding at which point it makes sense to provide recommendations for a place and how to manage change in management on the same location is another open research topic whose solution would greatly affect the quality of the suggestions that can be provided to customers, and as such improve the customer experience.

In terms of the current version of the application, the findings of the above performance and usability studies are encouraging regarding the capability of Planfree to scale beyond the current limited user basis: technically, MapReduce and Google App Engine provide for the necessary computing resources in a flexible and per-use basis that allows us to lock (and pay for) resources only if they are really needed. Content-wise, it seems that Planfree has the expected potential to appeal especially to locals in their own town (next to the personalized, purpose-specific recommendations, the presence of coupons and special offers seems crucial here), but the usability study also revealed a huge potential for the use of Planfree

when visiting another city the users are not yet familiar with (here the recommendation feature seems more important and the coupons a bit less).

The opportunities highlighted during the test will be used as guidelines for the upgrade of the interface of the platform, to make it even more easy and intuitive to use. In particular, we will improve the communication of the recommendation feature, making more clear the fact that the recommended restaurants are personalized and based on collaborative filtering.

The participants also produced a good list of missing features that we will consider for the next version of the platform. We will start from the “my favourite restaurants” to bookmark the places the user wants to remember or is planning to visit. The booking feature was already in our mind as a future step and we see that it will complete the experience in our platform. Some participants complained about a missing place where all available coupons are listed, but we will not add such feature: at the moment only coupons for the restaurants the user would like are shown to motivate people to try new places (and share their experience about them) and we plan to make them even more “personalized”.

Coupons will be used by restaurateurs to attract influential people that, according to the tastes we learned, will like the place and will share positive feedback. We don’t want to become likegroupon, where people go only for getting discounts, but we want to use them to motivate people to try new places at a lower risk and as marketing campaigns. Also, one idea that came up already during the development of Planfree and that may become a core distinguishing feature compared to other couponing platforms is the idea to allow the restaurateurs to know about the presence outside of their restaurant of people that could potentially be interested in their restaurant and to further allow them to make instant/on-the-fly offers that are specifically tailored to the interests of the identified group of people and

valid only for them and within a very short interval of time (e.g., one hour). This would allow restaurateurs to attract guests in a last minutes fashion.

## 8.6 Conclusion

In this chapter we presented Planfree, a prototype of restaurant recommender service. It includes all the learnings we collected through the studies presented in the previous chapters:

**Data collection** It collects ratings through the 5-star rating scale and the ratings are purpose-based, i.e. there are 4 ratings per item: with tourists, romantic, with friends and lunch break. The users are distinguished as locals or tourists, according to the city they are interested in.

**Algorithm** Despite the best algorithm found is user-based collaborative filtering, it is not adequate for online recommendations, so we moved to cluster-based collaborative filtering. It has the advantage of computing clusters of users in advance, making the computation at request time faster even in case of a large dataset of ratings.

**Availability everywhere and anytime** The recommender service is planned for fast and easy usage even from mobile devices, but without forgetting the needs of desktop users.

**Attracting users** One way to advertise the platform and motivate people to try this new service have been identified in a couponing service, which has been implemented. Thanks to coupons, both users and restaurateurs are motivated to contribute to Planfree, the first ones sharing their feedback and the second ones providing complete and updated information about their restaurants.

The prototype of recommender service has been completed, but some questions remains open.

As we have seen, the application of recommendations in an online service posed an important requirement: the recommendation list should be computed very quickly as the user is willing to wait only few seconds to get it. The recommender algorithm should be very fast and scalable, meaning that its performance is not affected by the amount of users and ratings it has to consider in its computations. The algorithm we identified as the best in Chapter 6, i.e. user-based collaborative filtering, is not applicable in such online context, and we had to adopt another algorithm that produces recommendations of lower quality. We need to search for a fast and scalable algorithm able to build recommendations of quality similar or better than user-based collaborative filtering.

Each characteristic of the recommender service implemented in Planfree has been tested against TripAdvisor's generic restaurant recommendations, and we know that combining personalization, purpose-based ratings and opinions from locals we can obtain better results than this very popular service. On the other hand, a deeper comparison with other personalized recommender systems is still missing. For example, we evaluated the best collaborative filtering algorithm (on our dataset) against Foursquare's generic rank in Chapter 6, but Foursquare builds also personalized recommendations that could provide better results.

The effects of providing the couponing service has not been tested yet. Despite discounts are widely adopted to attract people and motivate them to use a new service, it would be interesting to investigate their effect in a restaurant recommender service.





# Chapter 9

## Conclusion

In this thesis we tackled the problem of recommending places for leisure activities. The huge amount of offerings available in a city nowadays make it difficult for people to know all of them and to identify the ones they are more interested in. Recommender systems can support people's choice of where to spend their leisure time. We focused on restaurants as main use case, but our findings can be applied to places for other leisure activities.

In the following we summarize the findings, discuss the limitations of our studies and indicate possible directions for future work.

### 9.1 Contributions

The contribution of this thesis is a multifaceted study of how to recommend restaurants on mobile devices with a special eye to locals. The study identifies two features that affect the quality of recommendations as perceived by locals, namely the opinions of friends and the purpose of going out for a meal. Instead of focusing on the algorithmic side of the recommendation problem, the work focuses its attention on the quality of data by identifying how to best collect ratings from users on mobile devices and collecting data that are purpose-specific. A performance comparison of a set of collaborative filtering algorithms using the collected dataset shows that with

purposefully collected data even standard, off-the-shelf algorithms are able to outperform commercial recommendation systems like TripAdvisor.

The detailed contributions of the work are:

- A literature review on discovery of places, item recommendation, and rating collection;
- A comparative analysis of different neighborhood selection methods on collaborative filtering, exploring the effect of friend relationships and taste similarity;
- Two user studies of the effect of purpose on the choice of a restaurant;
- A comparative analysis of off-the-shelf collaborative filtering algorithms applied to purpose-based ratings;
- A prototype implementation and a performance analysis of a novel crowdsourcing framework for collecting reliable ratings, supporting the collection of large datasets for further studies;
- A prototype implementation of a personalized restaurant recommender service, following the results of the presented studies.

## 9.2 Lessons Learned

From the presented studies we learned the following.

- *Friends have a special influence in people's opinion, and recommender systems can use these relationships between users to improve their results.* We found that the ratings of friends with similar tastes are the best for predicting the requester's opinion, against considering only taste or friendship. The friends were collected from Facebook, a social network where real friendships are mixed with relationships

of lower strength, and we expect the results to be even better if we would be able to collect only those relationships considered important during leisure time.

- *The purpose, i.e. whether going at the restaurant with tourists, the partner or friends or price/quality ratio (also identified with lunch break), influences both the opinion about a restaurant (i.e. the assigned rating) and the choice of the restaurant.* Purpose-based personalized recommenders provide recommendations of similar quality for all purposes (i.e. bringing tourists, bringing the partner, bringing friends and price/quality ratio), having always higher precision than generic recommenders like the considered average-based baseline and TripAdvisor. These purposes have been shown to influence people's opinion in some cities (Trento in Italy, Asunción in Paraguay and Tomsk in Russia) and in different contexts. The purpose-based ratings collected allowed us to obtain different ranks for other activities other than going to the restaurant, such as going to a bar for aperitif, to a pub for a beer or to a club.
- *The most satisfying rating scale for rating items through mobile devices is 5-star, preferred over 3-faces, 3-thumbs and 2-thumbs.* 5-star provides users the needed granularity for correctly expressing their opinion and keeping the options easy to select even using small screens with touch interface, i.e. mobile devices.
- *The most precise personalized recommendations on mobile devices, providing the highest satisfaction to users, are obtained with user-based collaborative filtering, as compared with other off-the-shelf collaborative filtering algorithms.* This algorithm works particularly well for mobile recommender systems, where limited attention and small real

estate require short recommendation lists, such as the evaluated top-5 recommendations.

- *TripAdvisor's restaurant rank is different from locals' opinion.* Such difference was identified in the comparisons of Trento's locals opinions of four different datasets with TripAdvisor restaurant rank. Two of these datasets contain simple user-restaurant ratings and show that generic opinion of locals differs from TripAdvisor rank. In the other two datasets, with the introduction of purpose-based ratings, we were able to better understand the reason for this difference: according to locals, TripAdvisor is providing good recommendations for tourists, which is its goal, and for going out with the partner, but these recommendations are not good for going out with friends and according to price/quality ratio.
- *A novel crowdsourcing platform supports researchers and developers in collecting datasets of ratings efficiently.* Volunteering usually requires a high effort for motivating participants, while the proposed crowdsourcing platform can collect even more reliable and balanced ratings with lower effort, thanks to techniques for identifying cheaters (still respecting the subjectivity of workers' answers) and for assigning the best item to rate.

Throughout our research work, we learned that when building a recommender system it is important to keep focus on users: the system has to think in the same way as the requesters do, being able to recognize their tastes, considering the characteristics of the different items expressed through other users' ratings and identifying the best items as the requesters would do if they had enough knowledge and time to analyse all the items.

We learned that people needs can change over time, so different requests from the same user could need different results. These needs can

be influenced by the context of the request, and identifying what to consider as context is fundamental to be able to understand user needs and adapt results to them. In our studies we identified four purpose categories that showed to influence requester needs and more studies could be run to identify other contextual information that may influence people's choice of places for leisure activities.

### 9.3 Limitations

Such interesting work has though some limitations.

- The datasets we were able to collect for our studies have limited size, with the bigger including 162 users and 9,800 ratings. Despite we advertised our data collection, inviting also many colleagues and friends to participate, people were not willing to disclose their opinion as they were not receiving any direct benefit (such as high-quality recommendations) for their participation. In particular, from the user evaluation of rating scales and recommender algorithms (in Chapter 6) we perceive that some of the results could be significant, but the limited number of participants to the study does not let us make strong statements. In other cases available datasets, such as the ones provided by MovieLens and Netflix (with the smaller having 100,000 ratings), are used to avoid collecting novel datasets, even though movies are not the focus of their work.
- In our studies, we analyzed the personalized algorithms from the points of view of precision and satisfaction, but scalability and applicability in an online service are important too. Precision measures prediction accuracy and satisfaction measures how recommendations are perceived by people and how useful they are, while the time needed

to compute recommendations and the scalability of the service indicate how fast these results can be retrieved by users. Since people are not willing to wait more than few seconds, in particular when making the request on the go through mobile devices, the recommendations should be computed very quickly, even when the service grows and a huge amount of ratings need to be analysed.

- We focused on off-the-shelf recommender algorithms, while more advanced and precise algorithms have been developed. These algorithms have been used as they are popular and simple, letting us better communicate the effects of data.
- Comparisons of TripAdvisor's restaurant rank and locals' opinions are based on the external behaviour of TripAdvisor since its internal algorithm and dataset are not publicly available. The same holds for the comparison with Foursquare: only generic rank has been used while the service is able to provide personalized recommendations too.
- The proposed crowdsourcing platform for collecting reliable ratings has been tested against lazy and malign workers. Lazy workers assign ratings randomly to complete the rating tasks as fast or as effortless as possible, while malign workers give misleading ratings to particular items in order to reduce or raise their average ratings. Despite both of these types of workers are correctly marked as cheaters, the malign workers are able to pass the verification test as they can replicate their misleading ratings. A different verification test need to be identified for making the framework resistant to such malign workers, but still keeping it simple to pass for non-malign workers expressing a different opinion.

## 9.4 Future Work

TripAdvisor dataset and our purpose-based dataset of ratings provided by locals have many differences. While in TripAdvisor users rate usually few items in a city and, possibly, at different times, in our datasets locals assign many ratings in short time, making in this way a sort of personal rank. It would be interesting to measure how this different behaviour influences the quality of the produced recommendations or whether locals have a better knowledge of the different opportunities available than tourists and can better evaluate each single restaurant.

Our recommender system is able to perform better than TripAdvisor in recommending restaurants to locals, thanks to personalization, providing results of slightly better quality than the generic (i.e. non-personalized) Foursquare rank. Since Foursquare is able to provide personalized recommendations too, another comparison considering both personalization and purpose-orientation for Foursquare and Planfree is needed to better understand the strengths and weaknesses of these two services. We expect Foursquare to be stronger in personalization as its dataset contains more information about restaurants and users' past locations, while Planfree has a deeper knowledge of the different purposes.

We extended some off-the-shelf recommender algorithms to consider also purposes, making them able to analyse purpose-based ratings. The recommendations our service can compute could be improved by adapting more advanced and precise recommender algorithms. Moreover, it would be interesting to extend our studies by analysing other activities and identifying the more influencing purposes for them.

Different kinds of social and trust relationships, other than Facebook friends, are available through online social networks such as Twitter and Google+, and it would be interesting to study their effect on recommen-

dations. Google+ let us obtain different groups of people with different relationships with the user and could possibly let us identify the best group of friends for performing a specific activity along, while Twitter represents wider trust relationships and could let us identify those people that influence the user more.

As part of Planfree we presented the couponing service. It is supposed to motivate both users and restaurateurs to use the service, providing more feedback and high-quality information. Despite couponing is highly used as marketing strategy, we did not study its effects in the adoption of Planfree instead of its competitors and in quality and amount of ratings collected from the users.

A deep analysis of recommender algorithms and how to make their prediction more precise has already been done. Now there is a need for a deeper understanding of the way in which people usually choose a restaurant for their leisure time, identifying the information that influences their decision, related to the past experiences and the actual context. We have identified the role of purpose in the choice of restaurants, but more deeper studies on the way in which people choose a restaurant are needed. The more we learn about what influences people choices, the better we will be able to integrate similar considerations in recommender systems, making them able to replicate people's choices with more accuracy.

Another important aspect of recommender systems of places for leisure activities that would be interesting to explore more is their locality. Not only can people experience only items relatively close from a geographic point of view, but also the different culture and people behaviour in different countries could require the system to consider some information in a different way. For example, when the system receives the request for a recommendation of restaurants in December in Rio de Janeiro, Brazil, it has to make different considerations than for the same request in Tomsk,



Russia. In fact, in Brazil the weather is hot enough for eating outside, while in Russia a sit inside is needed as outside is very cold. This is just a simple difference dictated by the weather, but the different culture of people in such different countries could require different adjustments in the consideration of contextual information as people make choices in a different way. More studies about the different behaviour of people in different countries are needed to deeper understand such differences.



# Bibliography

- [1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.
- [2] Omar Alonso, Daniel E. Rose, and Benjamin Stewart. Crowdsourcing for relevance evaluation. *SIGIR Forum*, 42(2):9–15, 2008.
- [3] Taiwo Amoo and Hershey H Friedman. Do numeric values influence subjects’ responses to rating scales? *Journal of International Marketing and Marketing Research*, 26:41–46, 2001.
- [4] Lars Backstrom, Eric Sun, and Cameron Marlow. Find me if you can: improving geographical prediction with social and spatial proximity. In *Proceedings of the 19th International Conference on World Wide Web (WWW)*, pages 61–70, 2010.
- [5] Linas Baltrunas, Bernd Ludwig, Stefan Peer, and Francesco Ricci. Context-aware places of interest recommendations for mobile users. In *Design, User Experience, and Usability. Theory, Methods, Tools and Practice (DUXU)*, pages 531–540. Springer, 2011.
- [6] Philip Bonhard, Clare Harries, John McCarthy, and M. Angela Sasse. Accounting for taste: using profile similarity to improve recommender

- systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 1057–1066, 2006.
- [7] Fidel Cacheda, Víctor Carneiro, Diego Fernández, and Vreixo Formoso. Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems. *ACM Transactions on the Web (TWEB)*, 5(1):2, 2011.
- [8] Jonathan Chang and Eric Sun. Location3: How users share and respond to location-based data on social networking sites. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM)*, 2011.
- [9] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and Edward Y Chang. Parallel spectral clustering in distributed systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(3):568–586, 2011.
- [10] Dan Cosley, Shyong K Lam, Istvan Albert, Joseph A Konstan, and John Riedl. Is seeing believing?: how recommender system interfaces affect users’ opinions. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 585–592. ACM, 2003.
- [11] Henriette Cramer, Mattias Rost, and Lars Erik Holmquist. Performing a check-in: Emerging practices, norms and ‘conflicts’ in location-sharing using foursquare. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (Mobile HCI)*, pages 57–66. ACM, 2011.
- [12] Paolo Cremonesi, Franca Garzotto, Sara Negro, Alessandro Vittorio Papadopoulos, and Roberto Turrin. Looking for “good” recommenda-

- tions: A comparative evaluation of recommender systems. In *Human-Computer Interaction–INTERACT*, pages 152–168. Springer, 2011.
- [13] Alexander P. Dawid and Allan M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.
- [14] Munmun De Choudhury, Moran Feldman, Sihem Amer-Yahia, Nadav Golbandi, Ronny Lempel, and Cong Yu. Automatic construction of travel itineraries using social breadcrumbs. In *Proceedings of the 21st ACM conference on Hypertext and Hypermedia (HYPERTEXT)*, pages 35–44. ACM, 2010.
- [15] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer, 2011.
- [16] Carsten Eickhoff and Arjen P de Vries. Increasing cheat robustness of crowdsourcing tasks. *Information Retrieval*, 16(2):121–137, 2013.
- [17] Mojisola Erdt, Florian Jomrich, Katja Schüller, Christoph Rensing, et al. Investigating crowdsourcing as an evaluation method for TEL recommender systems. In *ECTEL meets ECSCW 2013: Workshop on Collaborative Technologies for Working and Learning*, page 25, 2013.
- [18] Amy Fong. The influence of online reviews. *Journal of Digital Research & Publishing Semester 1 2010 (7pm class)*, pages 106–113, 2010.
- [19] Edward W. Forgy. Cluster analysis of multivariate data: Efficiency versus interpretability of classifications. *Biometrics*, 21:768–769, 1965.
- [20] Ron Garland. The mid-point on a rating scale: Is it desirable? *Marketing Bulletin*, 2(1):66–70, 1991.

- [21] Damianos Gavalas, Vlasios Kasapakis, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. A survey on mobile tourism recommender systems. In *Third International Conference on Communications and Information Technology (ICCIT)*, pages 131–135. IEEE, 2013.
- [22] Cristina Gena, Roberto Brogi, Federica Cena, and Fabiana Vernerio. The impact of rating scales on user’s rating behavior. In *User Modeling, Adaption and Personalization (UMAP)*, pages 123–134. Springer, 2011.
- [23] Georg Groh and Christian Ehmig. Recommendations in taste related domains: collaborative filtering vs. social filtering. In *Proceedings of the 2007 International ACM Conference on Supporting Group Work (GROUP)*, pages 127–136. ACM, 2007.
- [24] Paula Hardy. *Italy Travel Guide*. Lonely Planet, 2012.
- [25] Matthias Hirth, Tobias Hoffeld, and Phuoc Tran-Gia. Cost-optimal validation mechanisms and cheat-detection for crowdsourcing platforms. In *Fifth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pages 316–321. IEEE, 2011.
- [26] Chien-Ju Ho, Shahin Jabbari, and Jennifer W. Vaughan. Adaptive task assignment for crowdsourced classification. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.
- [27] Chien-Ju Ho and Jennifer W. Vaughan. Online task assignment in crowdsourcing markets. In *AAAI*, 2012.
- [28] Tzvetan Horozov, Nitya Narasimhan, and Venu Vasudevan. Using location for personalized POI recommendations in mobile environ-

- ments. In *International Symposium on Applications and the Internet (SAINT)*. IEEE, 2006.
- [29] Tobias Hofffeld, Michael Seufert, Matthias Hirth, Thomas Zinner, Phuoc Tran-Gia, and Raimund Schatz. Quantification of YouTube QoE via crowdsourcing. In *IEEE International Symposium on Multimedia (ISM)*, pages 494–499. IEEE, 2011.
- [30] Jeff Howe. *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business*. Crown Publishing Group, 1 edition, 2008.
- [31] Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. Quality management on Amazon Mechanical Turk. In *Proceedings of the ACM SIGKDD workshop on human computation*, pages 64–67, 2010.
- [32] Ingrid Jeacle and Chris Carter. In TripAdvisor we trust: Rankings, calculative regimes and abstract systems. *Accounting, Organizations and Society*, 36(4–5):293 – 309, 2011.
- [33] Manas Joglekar, Hector Garcia-Molina, and Aditya Parameswaran. Evaluating the crowd with confidence. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 686–694, 2013.
- [34] Stephen C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.
- [35] David R. Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal task allocation for reliable crowdsourcing systems. *Operations Research*, 62(1):1–24, 2014.
- [36] Aniket Kittur, Ed H Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Confer-*

- ence on Human Factors in Computing Systems (CHI)*, pages 453–456. ACM, 2008.
- [37] Yehuda Koren and Robert Bell. Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer, 2011.
- [38] John Le, Andy Edmonds, Vaughn Hester, and Lukas Biewald. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *SIGIR workshop on crowdsourcing for search evaluation (CSE)*, pages 21–26, 2010.
- [39] Jin Ha Lee and Xiao Hu. Generating ground truth for music mood classification using Mechanical Turk. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries (JCDL)*, pages 129–138. ACM, 2012.
- [40] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. A comparative study of collaborative filtering algorithms. *arXiv report 1205.3193*, 2012.
- [41] Daniel Lemire and Anna Maclachlan. Slope One predictors for online rating-based collaborative filtering. In *SIAM International Conference on Data Mining (SDM)*, volume 5, pages 1–5, 2005.
- [42] Hongwei Li, Bo Zhao, and Ariel Fuxman. The wisdom of minority: Discovering and targeting the right group of workers for crowdsourcing. In *Proceedings of the 23rd International Conference on World Wide Web (WWW)*, 2014.
- [43] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *Internet Computing*, 7(1):76–80, 2003.
- [44] Janne Lindqvist, Justin Cranshaw, Jason Wiese, Jason Hong, and John Zimmerman. I’m the mayor of my house: Examining why peo-



- ple use Foursquare—a social-driven location sharing application. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 2409–2418, 2011.
- [45] Fengkun Liu and Hong Joo Lee. Use of social network information to enhance collaborative filtering performance. *Expert Systems with Applications*, 37(7):4772–4778, 2010.
- [46] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. Springer, 2011.
- [47] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining (WSDM)*, pages 287–296, 2011.
- [48] Simon Meyffret, Lionel Médini, and Frédérique Laforest. Trust-based local and social recommendation. In *Proceedings of the 4th ACM RecSys workshop on Recommender systems and the social web*, pages 53–60. ACM, 2012.
- [49] Joana Miguéns, Rodolfo Baggio, and Carlos Costa. Social media and tourism destinations: TripAdvisor case study. *Advances in Tourism Research*, pages 26–28, 2008.
- [50] Lik Mui, Mojdeh Mohtashemi, and Ari Halberstadt. A computational model of trust and reputation. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS)*, pages 2431–2439. IEEE, 2002.
- [51] Lik Mui, Peter Szolovits, and Cheewee Ang. Collaborative sanctioning: Applications in restaurant recommendations based on reputation.

## BIBLIOGRAPHY

---

- In *Proceedings of the 5th International Conference on Autonomous Agents (AGENTS)*, pages 118–119, 2001.
- [52] Arun Narang, Anurag Srivastava, and Naga Praveen Kumar Katta. High performance offline and online distributed collaborative filtering. In *IEEE 12th International Conference on Data Mining (ICDM)*, pages 549–558, 2012.
- [53] Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil. An empirical study of geographic user activity patterns in Foursquare. In *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 70–573, 2011.
- [54] Peter O’ Connor. User-generated content and travel: A case study on Tripadvisor.com. *Information and Communication Technologies in Tourism*, pages 47–58, 2008.
- [55] Gavin Potter. Putting the collaborator back into collaborative filtering. In *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 3:1–3:4, 2008.
- [56] Pearl Pu, Li Chen, and Rong Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems (RecSys)*, pages 157–164, 2011.
- [57] Francesco Ricci. Travel recommender systems. *IEEE Intelligent Systems*, 17(6):55–57, 2002.
- [58] Francesco Ricci. Mobile recommender systems. *Information Technology & Tourism*, 12(3):205–231, 2010.
- [59] Paul Rompf, Robin B Dipietro, and Peter Ricci. Locals’ involvement in travelers’ informational search and venue decision strategies while

- at destination. *Journal of Travel & Tourism Marketing*, 18(3):11–22, 2005.
- [60] Senjuti Basu Roy, Ioanna Lykourantzou, Saravanan Thirumuranathan, Sihem Amer-Yahia, and Gautam Das. Crowds, not drones: Modeling human factors in interactive crowdsourcing. In *DBCrowd 2013-VLDB Workshop on Databases and Crowdsourcing*, pages 39–42, 2013.
- [61] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web (WWW)*, pages 285–295. ACM, 2001.
- [62] Benjamin Satzger, Harald Psailer, Daniel Schall, and Schahram Dustdar. Auction-based crowdsourcing supporting skill management. *Information Systems*, 2012.
- [63] Ralf Schenkel, Tom Crecelius, Mouna Kacimi, Thomas Neumann, Josiane Xavier Parreira, Marc Spaniol, and Gerhard Weikum. Social wisdom for search and recommendation. *IEEE Data Engineering Bulletin*, 31(2):40–49, 2008.
- [64] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, pages 210–217, 1995.
- [65] Amit Sharma, Mevlana Gemici, and Dan Cosley. Friends, strangers, and the value of ego networks for recommendation. *Proceedings of the 5th International AAAI Conference on Weblogs and Social Media (ICWSM)*, pages 721–724, 2013.

- [66] Aaron D Shaw, John J Horton, and Daniel L Chen. Designing incentives for inexpert human raters. In *Proceedings of the ACM 2011 Conference on Computer Supported Cooperative Work (CSCW)*, pages 275–284. ACM, 2011.
- [67] Rashmi R Sinha and Kirsten Swearingen. Comparing recommendations made by online systems and friends. In *DELOS workshop: personalisation and recommender systems in digital libraries*, 2001.
- [68] E Isaac Sparling and Shilad Sen. Rating: How difficult is it? In *Proceedings of the fifth ACM conference on Recommender systems (RecSys)*, pages 149–156, 2011.
- [69] Yuandong Tian and Jun Zhu. Learning from crowds in the presence of schools of thought. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 226–234, 2012.
- [70] Beatrice Valeri, Marcos Baez, and Fabio Casati. Comealong: Empowering experience-sharing through social networks. In *Collaborative Computing: Networking, Applications and Worksharing (Collaborate-Com), 2012 8th International Conference on*, pages 175–180. IEEE, 2012.
- [71] Beatrice Valeri, Marcos Baez, and Fabio Casati. Come Along: Understanding and motivating participation to social leisure activities. In *Cloud and Green Computing (CGC), 2013 Third International Conference on*, pages 211–218. IEEE, 2013.
- [72] Beatrice Valeri, Florian Daniel, and Fabio Casati. D5.3 Planfree - Concepts, Architecture, Implementation and Exploitation. Technical report, TrentoRise, 2015.

- [73] Beatrice Valeri, Florian Daniel, and Fabio Casati. On the value of purpose-orientation and focus on locals in recommending leisure activities (submitted). *IEEE Internet Computing*, 2016.
- [74] Beatrice Valeri, Florian Daniel, and Fabio Casati. Rating scales and algorithms for mobile recommender systems: The case of restaurant recommendations (submitted). *Software: Practice and Experience*, 2016.
- [75] Beatrice Valeri, Florian Daniel, Fabio Casati, Mafe de Baggis, and Filippo Pretolani. D5.1 Recommendation and Emotional Representation of Places and Events: State of the Art. Technical report, TrentoRise, 2014.
- [76] Beatrice Valeri, Shady Elbassuoni, and Sihem Amer-Yahia. Crowdsourcing reliable ratings for underexposed items (submitted). In *Proceedings of the 12th International Conference on Web Information Systems and Technologies (WEBIST)*, 2016.
- [77] Daniel Gallego Vico, Wolfgang Woerndl, and Roland Bader. A study on proactive delivery of restaurant recommendations for Android smartphones. In *ACM RecSys Workshop on Personalization in Mobile Applications (PEMA)*, 2011.
- [78] Chirine Wolley and Mohamed Quafafou. Scalable expert selection when learning from noisy labelers. In *Proceedings of the 12th International Conference on Machine Learning and Applications (ICMLA)*, pages 398–401, 2013.
- [79] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th International*

## BIBLIOGRAPHY

---

*ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 114–121, 2005.

- [80] Kirsty Young. Social ties, social networks and the Facebook experience. *International Journal of Emerging Technologies and Society*, 9(1):20–34, 2011.