

Context-Aware Collaborative Filtering Recommender Systems

PHD THESIS IN COMPUTER SCIENCE

Linas Baltrunas

April 2011

Faculty Advisor: Prof. Francesco Ricci

Approved by: Prof. Maurizio Lenzerini, Prof. Giovanni Semeraro, Dr. Sergio Tessaris

Keywords: Recommender Systems, Context Aware, Collaborative Filtering, Algorithms, Experimentation

ACM categories: H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval – Information filtering, H3.4 [Systems and Software]: Performance evaluation – efficiency and effectiveness, G3 [Probability and Statistics]: Correlation and regression analysis, G1.6 [Optimization]: Gradient methods

Copyright © 2011 by Linas Baltrunas

Abstract

Recommender systems are intelligent tools that help on-line users to tame information overload. Most of the current recommender systems assume that users' preferences do not change quickly and previously recorded observations about what users like or dislike can help in predicting their future choices. This assumption is valid only to some extent. In fact, users' general interests can be relatively stable, but they can also be influenced by many additional and varying factors, here referred as context. For example, weather could play an important role while choosing between going to a beach or visiting an indoor museum.

In the thesis I have studied how contextual information could be exploited to improve the prediction accuracy of Recommender Systems based on the Collaborative Filtering technique. Within this thesis we analyzed several new methods and techniques to perform context-aware Collaborative Filtering. We have also introduced and investigated a comprehensive development methodology for context-aware system. It consists of a two steps data gathering process and a specialized model-based rating prediction algorithm. We validated this methodology on a mobile recommender that provides real-time context-aware recommendations and can explain such recommendations on the base of the acquired contextual information. We have shown in an experimental study that our context-aware methods can increase the prediction accuracy compared to some context-free approaches and other state of the art context-aware methods. In addition, we carried out a user study that showed that the proposed techniques can also increase the user satisfaction.

Acknowledgments

I want to thank all the people that helped me to start and complete the theses. First of all I am grateful to my advisor Francesco Ricci who was also my mentor through all these years. He introduced me to the field of Recommender Systems, kept me motivated and guided through the interesting and enriching path of the research. It was a big pleasure and benefit to work with such a strong scientific advisor and a great personality.

I am also thankful to the whole faculty of Computer Science and especially to the members of DIS group. During the master studies they established my background for doing research and kept me growing since then. I also want to express my thanks to all the people with whom I worked together during these years. Special credit goes to the Multimedia group of Telefónica and especially to Xavier Amatriain, Alexandros Karatzoglou and Nuria Oliver with whom I did research on the Tensor Factorization presented here. Bernd Ludwig who visited our faculty greatly contributed with his ideas and expertise to the ReRex project. My student Stefan Peer worked with me on the graphical user interface of ReRex and helped me to start developing my teaching abilities. I also want to thank my external reviewers - Prof. Giovanni Semeraro and Dr. Sergio Tessaris. I am very pleased for their comments and help in improving the thesis.

Besides the support in the research, I also want to thank my whole family. They were the first to nudge me into the research and kept me motivated with their interest in my work and their warm atmosphere. Moreover, they themselves were great examples of scholars and scientists.

And of course my biggest thanks to all my friends and closest people to me. We shared so many great experiences and thoughts that will always stay with me.

Contents

Abstract	iii
Acknowledgments	v
I Introduction	1
1 Motivation and Contributions	3
1.1 Research Motivation	3
1.2 Context	5
1.3 Motivating Examples	7
1.4 Contributions	8
1.4.1 List of Publications	11
1.5 Organization of the Thesis	13
2 State of the Art	15
2.1 Defining Context	16
2.2 Recommender Systems	18
2.3 Historical Perspective of Context Aware Collaborative Filtering .	21
2.4 Context-Aware Mobile Tourist Guides	27
II Algorithms for CACF	31
3 Item Splitting	33
3.1 Method	35
3.1.1 Splitting Criteria	36
3.1.2 Complexity of the Algorithm	38
3.1.3 Missing Context Values	39
3.2 Experimental Evaluation	40

3.2.1	Performance on Contextually-Tagged Data	42
3.2.2	Evaluation on Data containing Demographic Information	44
3.2.3	Varying the Impact of the Contextual Variables	47
3.2.4	Changes in Recommendation List	48
3.2.5	Performance with Missing Contextual Values	50
3.2.6	Precision and Recall Analysis	50
3.3	Conclusions and Future Work	54
4	N-dimensional Tensor Factorization	57
4.1	Related Work	58
4.2	Multiverse Recommendation	59
4.2.1	Matrix Factorization	60
4.2.2	Tensor Factorization	60
4.2.3	Tensor Factorization for CF	62
4.2.4	Loss Function	62
4.2.5	Regularization	63
4.2.6	Optimization	64
4.2.7	Missing Context Information	65
4.3	Experiments	66
4.3.1	The data	66
4.3.2	Evaluation Protocol	66
4.3.3	Alternative Context-based Methods	67
4.4	Results	68
4.4.1	Tensor vs. Matrix Factorization	68
4.4.2	Comparison to Context-Aware Methods	69
4.5	Conclusions	70
5	Group Recommendations	73
5.1	Rank Aggregation	76
5.2	Experimental Setup	78
5.2.1	Data Set and Group Generation	79
5.2.2	Recommendation Lists for a Group	80
5.2.3	Effectiveness of a Ranked List of Recommendations	80
5.3	Experimental Results	81
5.3.1	Effectiveness of Group Recommendations	82
5.3.2	Group Versus Individual Recommendations	84
5.4	Discussion and Conclusions	86
6	Item Weighting and Item Selection	87
6.1	Adaptive Similarity	90
6.1.1	Weighting Schemas	90
6.1.2	Weight Incorporation	93

6.1.3	Item Selection	94
6.2	Experimental Evaluation	96
6.2.1	Evaluation of Weight Incorporation	98
6.2.2	Evaluating Weighting Schemas	99
6.2.3	Evaluating Item Selection	101
6.3	Discussion and Conclusions	104
III	Implementing CACF	107
7	Building Context-Aware Collaborative Filtering Systems	109
7.1	Challenges of Building CACF System	109
7.1.1	Context-Aware Recommendations are Difficult to Compute	110
7.1.2	Towards a Methodology for Developing CARS	110
7.1.3	ReRex: The New Methodology in Practice	112
7.2	Context Dependent Data Acquisition	113
7.2.1	Overview of the Implemented Data Acquisition Process. .	114
7.3	Acquiring Context Relevance	115
7.3.1	Analysis of Context Relevance	117
7.4	Acquisition of Ratings in Context	119
7.5	Context-Aware Recommendation Algorithm	122
7.5.1	The Predictive Model	123
7.5.2	Model Training	126
7.6	Context-Aware Place of Interest Recommendations	131
7.7	Evaluation	135
7.7.1	Usability Test	137
7.8	Conclusions and Discussion	140
IV	Conclusions and Summary	143
8	Conclusions and Future work	145
8.1	Summary and Conclusions	145
8.2	Future Work	147
A	Symbol Table	151
B	Ranking of Contextual Factors	153
B.1	How Does Context Influence the Decisions of Users?	154

C Data Sets	155
C.1 Data Sets Used in Off-Line Evaluation	155
C.1.1 Adom	155
C.1.2 Food	156
C.1.3 Yahoo! Webscope	157
C.1.4 Yahoo! Semi-Synthetic Data Sets	157
C.1.5 Movielens	158
C.1.6 Bolzano POIs	158
Bibliography	159

Part I

Introduction

Motivation and Contributions

1.1 Research Motivation

Recommender Systems (**RS**) are powerful tools helping on-line users to tame information overload by providing personalized recommendations on various types of products and services. Internet interconnects millions of on-line services offering a huge amount of various kinds of information, products and experiences. More notably, a single e-commerce web site can offer up to hundreds of thousands items in different categories and for different types of users. Not surprisingly, users, who are looking for the next book to read, or a digital camera to buy, are overwhelmed by the quantity of information to consider. Hence, they may find difficult to filter out the irrelevant information and choose the most suitable products without some assistance tailored to the specific needs and knowledge of the user. To address these problems recommender systems have been proposed and now are largely diffused. These are intelligent applications that are able to identify and suggest the products, information or services that best fit the user needs and preferences. Applications in many domains, including book recommendation, movie suggestions, travel service information and many other, have reported success stories of using recommender systems [Linden et al., 2003, Peddy and Armentrout, 2003].

Recommendations are usually computed by exploiting historical data of the users' online behavior [Adomavicius and Tuzhilin, 2005]. Assuming that the user's behavior does not change quickly then the previously recorded observations can help in predicting the future behavior. In fact, this assumption is valid only to some extent, as the user's general interests can be relatively stable, but, they can also be influenced by many additional and varying factors. For example, imagine a user who spends his week-ends actively. This aspect could be used to characterize the hobbies of the person, but the activity itself depends on

many other factors such as the current season, weather or companion. The vast majority of recommender systems fail to adapt to such changing situations, which can result in poor quality recommendations.

This thesis analyzes how these additional factors, here referred as **context**, could be exploited to improve the output of a Recommender System algorithm, i.e., its recommendations. Context Aware Recommender Systems (**CARS**) is an active and growing research area. It raises interesting and difficult problems related to the improvement of the existing approaches in order to model and exploit new sources of dynamically changing data. In fact, the interest comes not only from academia but also from industry. During my Ph.D. studies I took part in a research project of Telefónica Investigación y Desarrollo, where we studied model based techniques for CARS [Karatzoglou et al., 2010]. Moreover, I started there a research on using implicit data with context [Baltrunas and Amatriain, 2009]. At the moment of writing theses the obtained results were not conclusive, therefore, are not reported in the Theses. We are continuing the research line and switch to using Tensor Factorization techniques for implicit data. The other on-going research project is sponsored by Deutsche Telekom. Here we are investigating new techniques to acquire context-dependent preference data [Baltrunas et al., 2010a] and analyzing context aware recommendations in the music domain.

Context is a multifaceted concept that has been studied across different research disciplines [Adomavicius and Tuzhilin, 2010]. In our work we adopt the definition by Dey: *“Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application.”* [Dey, 2001]. A Recommender System can consider as contextual information, e.g., in a travel planning application, the companion, the weather, the temperature or the location. For other definitions of context please refer to the State of the Art Chapter on page 15. We note also that we employ the “representational” view of context [Dourish, 2004]. According to this view, contextual dimensions and their possible values are listed a-priori by system designers. Such view gives us a simple and intuitive definition that can be used to build accurate predictive models. In this thesis we consider only nominal values of the context such as weather is “sunny”, or the user companion is a “colleague”. The typical data that we collected and used in our work consists of subjective user ratings that are recorded together with the (available) values of the variables representing contextual conditions. For example, a user gave rating 5 for the Messner Mountain Museum when it was sunny and he was alone.

Within this thesis we are aiming at two main objectives. Firstly, using context-dependent preference data we want to generate more relevant recommendations. We have measured recommendation relevance mostly by means of off-line experiments, but we have also been conducting a user study to show how contextual information can increase the user satisfaction with the gener-

ated recommendations. Secondly, when making a recommendation, we claim that its effectiveness is largely dependent on the user's perception of how the various knowledge sources (providing information on the user, on the item, or on contextual information) impact on the recommendations. Therefore, we also used contextual information to make clearer and more transparent this effect, by actually explaining the provided suggestions.

To achieve these goals we have built recommendation algorithms on top of state of the art techniques in the field. We have adopted an existing data model for context representation and extended classical RS approaches to be used in a context-aware setup. The introduced methods were proved to be more accurate and scalable, compared to other state of the art solutions, using a well accepted off-line evaluation methodology and measures. Moreover, in order to assess the benefits of using context-awareness beyond the mentioned improvement of the recommendation accuracy, we built a prototype system that generates recommendations using one among the developed algorithms. This system was evaluated in a live users experiment, after context-enriched rating data were collected in order to bootstrap the recommender.

1.2 Context

To understand the importance of context in RS we need to look at the general problem of learning from data. Predictive models rely on the assumption that the data generated by a process or a system in the future will show similar patterns (distribution) to the data previously collected observing the same process or system [Mitchell, 1997]. In a RS this means that user ratings for items are consistent over time and we can use them to predict future ratings for yet unseen items.

Recommender systems usually model the user preference for an item with a single rating. A rating can mean different things for the user: his general preferences for an item, likelihood of purchasing the item, likelihood to consume the item. To illustrate this issue, we adapt an example made by F. Ricci in his invited speech at CAMRA workshop [Berkovsky et al., 2010]. Imagine that you rate cars on the imaginary “dream cars of the year” website. Most users would give highest ratings for the most impressive cars. But, it this would be different if the user would be asked to give a rating that reflects his/her intention to buy any of these cars. Now, depending on the user, most expensive cars could get much lower ratings, despite that they are impressive cars, i.e., they have high ratings “as dream cars”.

In several simple domains, a single rating for a user-item pair works fine. But in the majority of the application domains for recommender systems various types of contextual conditions are relevant as much as the stable preferences of

the user, e.g., I like science fiction and classical music. In fact, we read different books in different periods of the year or of the day, we listen different music when we are working or having a party, we watch different movies if we are alone or with our family. All these examples illustrate that the context does matter for the recommendation.

But still, the majority of the RSs do not *explicitly* model and take into account contextual information. This does not mean that context is not present. All RS assumes the *default context* defined by the system itself or established by the users of the system. For example, when a user comes to a movie recommender site the big part of context is established. The domain is known, user task and goals are also known. Moreover, the interpretation of the ratings is also provided. Usually the user is instructed to provide a personal rating reflecting how much he liked the movie. In most situations this simplified setting is adequate and still accurate recommendations can be provided. This includes domains where items are usually used only in very standard situations, or the alternative situations themselves have no major impact on the personal experience of the recommended items. The problem arises with those domains where items can be consumed in alternative situations that have a large impact on the experience. For example, weather could play important role while choosing between going to a beach or visiting an indoor museum.

In a travel planning application context could be the companion, the weather, the season or the location. More technically, we define context as a *dynamic* set of features C that describe the state of the user, the item, the user experience, and that can *rapidly* change their values. Context depends on the user and on the item, or even the combination of both. Context $c = (c_1, \dots, c_k), c_j \in C_j$ describes the conditions under which the user experience (rating) was collected and the situation of the target user asking for a recommendation. The subjective rating r_{uic} given by a user u for item i is therefore recorded together with the state of the contextual conditions c at the time the rating was provided. For instance, consider an example where three contextual dimensions are given: weather $C_1 = \{sunny, cloudy, rainy, snowing\}$, user mood $C_2 = \{happy, sad\}$ and crowdedness $C_3 = \{crowded, not - crowded\}$. This applies to the recommendation for places of interest (POI). Then when the user rates a POI a typical CARS will store this rating together with the context $C_1 \times C_2 \times C_3 \ni c = [sunny, happy, crowded]$ under which the item was experienced and hence the rating provided. When making recommendations, the context of the request is used for query refinement, i.e., a context-aware recommender system tries to retrieve the most relevant items for a user, given the knowledge of the context of the request. We note that in some cases it could be hard to establish a sharp separation between context and item or user description. For instance, the mood of the user can be described as a contextual condition for a particular request or a property of the user (user profile). In general, the dynamic aspect of context is very important.

For example, we consider mood a contextual dimension of the user as it can change its state from moment to moment. A user can be happy in the morning and sad in the evening. On the other hand we do not refer to his nationality as context because it is stable and hardly ever changes.

Note, that contextual dimension C_i can take a range of nominal values. We consider only the nominal values mainly because of the simplicity reasons. Supporting ordinal, real, tree-structured values could provide additional domain knowledge and improve prediction accuracy but would also make the proposed solutions more complicated. Nevertheless, some of the proposed algorithms could be easily extended to support specific values, however, other methods do not have a trivial solution. For example, all of the proposed CACF algorithms could be easily adapted to support the real value domains by using standard discretization techniques [Han et al., 2006]. Item-splitting (see Chapter 3) method could be extended to support ordinal domains by constraining the splitting search space. However, it is not clear how to use order information in factorization techniques described in Chapter 4 and Chapter 7. Moreover, tree-structured values could be integrated in the factorization techniques by treating each level of the tree structure as a separate dimension of context. However, in order to have a unified data model in this Thesis we use only nominal values, and we will address the issues related to continuous values in our future work.

1.3 Motivating Examples

This section provides motivating examples from a tourism domain. We will build a scenario which will point out the benefits of CARS. Later we will give several examples, where RS would clearly benefit from context-awareness.

Imagine Bob to be a tourist, visiting Bolzano city. He is 32 years old, single, computer literate, keen on hiking and he is interested in contemporary art. It is his first time in Bolzano, the gateway to the world famous Dolomite mountains. Even if Bob carefully planned his mountain trip in the Dolomites, he had no time to take a better look at what to do in Bolzano. He arrives at noon to his hotel in the city center and starts planning what to do next. Bob is keen on stretching his legs before departing to the higher grounds. After browsing the Web on his mobile device he finds a couple of walking paths around the city. It is not comfortable to browse the Internet on the mobile phone, especially, when there are so many possibilities to choose from. Therefore, without too much care Bob chooses the Sant’Osvaldo Promenade. Just at the base of the mountain, where the path begins, sudden rain starts. Storms are approaching so fast in the mountain region. The promenade is out of the question - but what to do now?

Now imagine that Bob has installed on his iPhone ReRex - a mobile context-aware Recommender System. The detailed application description can be found in

Chapter 7. The application is endowed with a prediction model that was trained using the explicit ratings collected from a user population. It is able to take into account, not only these ratings data, but also weather and other contextual conditions such as season, temperature, budget, etc. When the application is started, the user is presented with six non-personalized recommendations for his current location in Bolzano city. The user can specify his preferences for categories of POIs in order to get more relevant recommendations. The list is automatically refreshed, while pulling the current weather information from the near-by airport. The real-time weather information forecasts sudden rain, therefore the suggestion list is automatically adapted. All the recommendations for walking paths are replaced by more relevant suggestions for indoor activities such as South Tyrol Museum of Archaeology and Museion - Museum of Modern and Contemporary Art. The current recommendations are explained: “This place is good to visit even on a rainy day”. Bob appreciates the exploitation of the weather forecast information. Bob changes his initial plan, i.e., to go for a walk. He is also interested in contemporary art, thus follows the recommendation to visit Museion by adding it to the wish list.

Clearly, above we illustrated only a simple scenario, where weather context is important for filtering out irrelevant recommendations. We can imagine other contextual dimensions, which should as well be taken into account to make more relevant recommendations. The **season** is important for certain types of places of interest. For example, in summer a hiking place in Obereggen mountains near Bolzano could become skiing place in winter. Classical recommender systems would not distinguish these differences, as an item (Obereggen mountain) would be represented with a single set of ratings. The **weather** could play an important role while recommending beaches. It makes no sense to recommend sunbathing in a beach while it is raining. Similarly **temperature** could be important for choosing between outdoor and indoor museum. **Available amount of time** should be taken into account while recommending items that usually take a lot of time to visit. For example, if a person has only a couple of hours available, it is not worth recommending a hiking path which takes full day. **Budget** has influence while selecting expensive attractions, such as paragliding or canoeing. Clearly, the **distance** to the place of interest plays a very important role. It is wrong to recommend architecture masterpieces of Gaudi in Barcelona if the user is in Bolzano and has only half a day for a visit. Other examples of context, that we used in a built prototype can be found in Table 7.1 on page 116.

1.4 Contributions

This section describes the main results of this thesis. We start by shortly describing the main achievements and finish with the list of publications.

New algorithms improving the accuracy and scalability of Context Aware Collaborative Filtering We have introduced and experimentally tested a number of algorithms for CACF. The summary of the proposed methods with their advantages and disadvantages is shown in Table 1.1.

Technique	Advantages	Disadvantages	References
Item Splitting	Easy to integrate with existing CF systems, intuitive heuristic.	Less accurate, needs selection of impurity measure.	Chapter 3, [Baltrunas and Ricci, 2009a] [Baltrunas and Ricci, 2009b], [Baltrunas and Ricci, 2010].
Linear Model	Constant time prediction; accurate; contains explanation heuristic.	Assumes independence of contextual dimensions (in some cases an advantage).	Chapter 7 [Baltrunas et al., 2011a].
Tensor Factorization	Accurate; exploits dependencies between contextual dimensions.	Expensive to train.	Chapter 4, [Karatzoglou et al., 2010]
Group	Exploits user models of companion dimension.	Just for companion dimension.	Chapter 5, [Baltrunas et al., 2010b].
Item Weighting	Extends well accepted memory based approaches.	Requires similarity measure between contexts, expensive predictions. Was not evaluated with context-enriched data.	Chapter 6, [Baltrunas and Ricci, 2009c], [Baltrunas and Ricci, 2007].
BIPO Item Selection	Extends well accepted memory based approaches.	Requires similarity measure between contexts, expensive predictions. Was not evaluated with context-enriched data.	Chapter 6, [Baltrunas and Ricci, 2008], [Baltrunas and Ricci, 2007].

Table 1.1: Contributed techniques and their characteristics.

In Chapter 3 we describe the Item-splitting approach that overcomes a number of limitations of the (current state of the art) reduction-based approach [Adomavicius et al., 2005]. Item-splitting reduces computational complexity and improves accuracy. In item splitting the set of ratings for an item are split into two subsets according to the value of a contextual variable, e.g., ratings collected in “winter” or in “summer” (here the contextual variable is the season of the rating/evaluation). These two sets of ratings are then assigned to two new fictitious items (e.g. beach in winter and in summer). This split is performed only if there is a statistical evidence that under these two contextual conditions the item’s ratings were different, i.e., users evaluate the item differently. This technique was published in a conference proceedings publication [Baltrunas and Ricci, 2009a] and some more details are described in two workshop papers [Baltrunas and Ricci, 2009b, Baltrunas and Ricci, 2010].

More recently, we have presented at RecSys 2010 conference a new technique [Karatzoglou et al., 2010] that exploits Tensor Factorization (TF) for CACF. To our knowledge, this is the first pure model based approach that incorporates several contextual dimensions into CF. It provides even more accurate predictions

compared to all our previously tested memory-based approaches. Tensor Factorization is a generalization of Matrix Factorization that allows for a flexible and generic integration of contextual information. In the proposed model different features of context are considered as additional dimensions in the representation of the data as a tensor. The factorization of this tensor leads to a compact model of the data which can be used to provide context-aware recommendations. More details about that can be found in Chapter 4.

New techniques for improving the explainability of Context Aware Collaborative Filtering We have designed a new approach for generating explanations based on a CACF prediction model. To our best knowledge this is the first time that context is used to motivate recommendations. To attain this goal we have developed a simple linear model that combines Matrix Factorization based CF and context biases into a single approach. The main advantage of this approach is its suitability in explaining the recommendations. Since the contextual conditions are linearly combined to determine the final prediction of the rating it is immediate to determine the conditions that mostly influence the rating value. We conjectured that these are the best conditions to present to the user, as motivations for the particular recommendation. This work was submitted to [Baltrunas et al., 2011a] and is currently under the review process. The details of the methods can be found in Chapter 7.

Data acquisition methodology for CARS We have proposed and evaluated a system design methodology for context-aware data acquisition, and recommender system bootstrapping. The results of this research was described in [Baltrunas et al., 2011b] and an extended version was submitted to the journal [Baltrunas et al., 2011a] which is currently under the review process. We also proposed an approach for collecting context-aware data that relies on the notion of “best context”. In this case instead of asking the user to rate items in context, we acquire information about the preferred context of usage for an item. This approach greatly reduces the quantity of data that should be acquired from the user to produce a context-aware recommendation. The results of this research were presented in a workshop [Baltrunas et al., 2010a]. The details of the acquisition methodologies can be found in Chapter 7.

Implementation and evaluation of system prototypes The techniques described above have been implemented and incorporated in two prototype systems. The first, called ReRex, is a context-aware system that recommends places of interest in Bolzano. The second, called InCarMusic, is a system prototype that we have built in a project funded by Deutsche Telekom. It generates context-aware, group-dependent music recommendations to be listened while driving a

car. The work is in progress and the results are not reported in the theses. We made a small scale user study, investigating the system usability, accuracy, and acceptance of these CARS. The graphical user interface design and evaluation of ReRex system was presented as a part of the bachelor thesis of Stefan Peer [Peer, 2010], whom I co-supervised with Prof. F. Ricci. The extended description of this research was submitted to [Baltrunas et al., 2011a]. The details of the system can be found in Section 7.5.

Group Recommendations Adapting recommendations to a group of users can be seen as a particular form of context adaptation. In [Baltrunas et al., 2010b], a research work published in RecSys '10, we have compared the effectiveness of individual and group recommendation lists using normalized discounted cumulative gain. We have discovered that the effectiveness of a group recommendation, when it is computed with rank aggregation methods, does not necessarily decrease when the group size grows. Moreover, when individual recommendations are not effective we have illustrated how a user could obtain better suggestions looking at the group recommendations. See Chapter 5 for the details.

Similarity Metric Adaptation in Memory Based CF In an earlier activity conducted during my PhD research we have proposed to use Item Selection in the neighborhood formation step of a memory based CF approach. The proposed item selection method consists of selecting the item ratings that are used to establish the user-to-user similarity, as a function of the specific rating that one must predict. This approach improves the accuracy of CF while using less information in making rating predictions. This result was presented in a workshop [Baltrunas and Ricci, 2007] and then further extended in a conference paper [Baltrunas and Ricci, 2008]. We also investigated item weighting and the results of this research were published as a book chapter [Baltrunas and Ricci, 2009c]. Item weights try to estimate how much a particular item is important for computing the rating predictions for a target item. These techniques could be extended to incorporate contextual information into the memory based CF approaches. We found that some approaches are better performing and showed that item weighting can always improve the accuracy of CF if these weighting methods are used. The description and evaluation of item weighting and item selection techniques can be found in Chapter 6.

1.4.1 List of Publications

In this subsection we give a list of the publications grouped by the type of publication and ordered by the date.

- conference level publications:

- Baltrunas, L., Ludwig, B., and Ricci, F. (2011). Context relevance assessment for recommender systems. In Pu, P., Pazzani, M. J., André, E., and Riecken, D., editors, *IUI*, pages 287–290. ACM.
- Baltrunas, L., Makcinskas, T., and Ricci, F. (2010). Group recommendations with rank aggregation and collaborative filtering. In *RecSys’10: Proceedings of the fourth ACM Conference on Recommender Systems*, pages 119–126, New York, NY, USA. ACM.
- Karatzoglou, A., Amatriain, X., Baltrunas, L., and Oliver, N. (2010). Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys’10: Proceedings of the fourth ACM Conference on Recommender Systems*, pages 79–86, New York, NY, USA. ACM.
- Baltrunas, L. and Ricci, F. (2009). Context-based splitting of item ratings in collaborative filtering. In Bergman, L. D., Tuzhilin, A., Burke, R. D., Felfernig, A., and Schmidt- Thieme, L., editors, *RecSys*, pages 245–248. ACM.
- Baltrunas, L. and Ricci, F. (2008). Locally adaptive neighborhood selection for collaborative filtering recommendations. In Nejdl, W., Kay, J., Pu, P., and Herder, E., editors, *AH*, volume 5149 of *Lecture Notes in Computer Science*, pages 22–31. Springer.
- workshop level publications:
 - Baltrunas, L. and Ricci, F. (2010). Context – dependent recommendations with items splitting. In Melucci, M., Mizzaro, S., and Pasi, G., editors, *IIR*, volume 560 of *CEUR Workshop Proceedings*, pages 71–75. CEUR-WS.org.
 - Baltrunas, L., Kaminskas, M., Ricci, F., Rokach, L., Shapira, B., and Luke, K.-H. (2010). Best usage context prediction for music tracks. In *2nd Workshop on Context-Aware Recommender Systems (CARS-2010)*.
 - Baltrunas, L. and Ricci, F. (2009). Context- dependent items generation in collaborative filtering. In Adomavicius, G. and Ricci, F., editors, *Proceedings of the 2009 Workshop on Context- Aware Recommender Systems (CARS-2009)*.
 - Baltrunas, L. and Amatriain, X. (2009). Towards time-dependant recommendation based on implicit feedback. In Adomavicius, G. and Ricci, F., editors, *Workshop on Context-Aware Recommender Systems (CARS-2009)*.
 - Baltrunas, L. and Ricci, F. (2007). Dynamic item weighting and selection for collaborative filtering. In Berendt, B., Mladenic, D.,

Semeraro, G., Spiliopoulou, M., Stumme, G., Svatek, V., and Zelezny, F., editors, Web Mining 2.0 International Workshop located at the ECML/ PKDD 2007, pages 135–146.

- book chapter:
 - Baltrunas, L. and Ricci, F. (2009). Item weighting techniques for collaborative filtering. In Berendt, B., de Gemmis, M., Semeraro, G., Spiliopoulou, M., Stumme, G., Svatek, V., and Zelezny, F., editors, Knowledge Discovery Enhanced with Semantic and Social Information, volume 220 of Studies in Computational Intelligence, pages 109–127. Springer Berlin / Heidelberg.
- journal publication:
 - Baltrunas, L., Ludwig, B., Peer, S., and Ricci, F. (2010). Building real-time context-aware recommendations for mobile users. Personal and Ubiquitous Computing. **(under review process.)**

1.5 Organization of the Thesis

The thesis is divided into four parts: Introduction (page 3), Algorithms (page 33) to perform Context Aware Collaborative Filtering (CACF), Implementation of the prototype system (page 109), and Conclusions (page 145).

In the introduction part we motivate the research, define the notion of context and list the contributions of this thesis. Chapter 2 gives an introduction to Recommender Systems and provides a historical perspective on the related work on Context Aware Collaborative Filtering.

The second part of the thesis gives details on our contributed algorithms for CACF. We start with the heuristic based Item- splitting (Chapter 3) and the model based Tensor Factorization (Chapter 4) methods. Then we talk about group recommendations (Chapter 5) which is a special case of context dimension. At the end of the third part we give details on the item weighting and item selection approaches (Chapter 6).

The third part of the thesis introduces the development cycle of a CACF Recommender System. We show a case study in the tourism domain, where we built a prototype for recommending places of interest in Bolzano city. This is a practical part where we analyzed data acquisition problem and provided analysis of our proposed solution. Then we move on to the analysis of our implemented prototype system called ReRex (see Chapter 7). The system includes our developed CACF algorithm and graphical user interface. We show the results of the user study that we carried out during the testing phase of the system.

The last part of the thesis summarizes the work, draws the conclusions and talks about future research directions.

State of the Art

This chapter overviews related work in the field of CARS. Here we will discuss approaches that actually use context to generate recommendations. The details on the underlying techniques (such as gradient based optimizations) will be provided if needed in the Chapters where we discuss their usage.

Recommender Systems (RS) is an active and fast growing field of research. In recent years many techniques for making recommendations were proposed. We will start this section with a short overview of the field (see Section 2.2) illustrating the major techniques and using a well established classification of RS approaches. After this general introduction to RS we will move on to the problem of context awareness. This was shown to be important across various application domains, such as advertising and economics (Section 2.1). Next, we will concentrate on context awareness in Recommender Systems, with the major focus on Collaborative Filtering. To understand better the evolution of this field we will follow an historical perspective starting from some early contributions dated 2001, and concluding with research presented at the 2nd Workshop on Context-Aware Recommender Systems in September 2010 (see Section 2.3). In the same Section we will list algorithms for Context Aware Collaborative Filtering (**CACF**), analyzing competing classes of methods and pointing out their strongest and weakest features. We will further explain how the research shifted towards approaches that use regression models, such as tensor factorization, to model context in CF. Then, after having illustrated algorithms for CARS we will overview the usage of context in a Tourism Domain (Section 2.4). This section is aimed at illustrating how specific could be the context dimensions in each domain.

2.1 Defining Context

Context is a multifaceted concept that has been studied across different research disciplines, including computer science (primarily in artificial intelligence and ubiquitous computing), cognitive science, linguistics, philosophy, psychology, and organizational sciences [Adomavicius et al., 2005]. Each discipline tends to narrow down the standard generic dictionary definition of context as “conditions or circumstances which affect some thing” [McKechnie, 1983]. Therefore, there exist many definitions of context across various disciplines and even within specific subfields of these disciplines. Bazire and Brézillon [Bazire and Brézillon, 2005] present and examine 150 different definitions of context from different fields.

Even within the context-aware computing literature context has a very broad definition. In early work Schilit and Theimer [Schilit and Theimer, 1994] define context as “location and the identity of nearby people and objects”. This definition was broadened by Schilit et al. [Schilit et al., 1994]: “Context encompasses more than just user’s location, because other things of interest are also mobile and changing. Context includes lighting, noise level, network connectivity, communication costs, communication bandwidth and even the social situation, e.g. whether you are with your manager or with a co-worker.” Only several years later Dey moved from a definition by examples to a more abstract definition: “*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application*” [Dey, 2001]. This is probably the first definition that was broadly adopted in the computational sciences. In RS, entity is usually a user, an item and the experience that user is evaluating. Dourish [Dourish, 2004] further analyzed the definition of context in computational environments. He distinguished two main views of context: *representational* and *interactional*. According to the author, the representational view tries to separate context from the action. In some sense context defines an action and provides some form of information about it. This view also assumes that context could be defined beforehand of an action and it has a single interpretation across various activities. On the opposite, the interactional view defines context as a relational property. Moreover, the scope of the context is defined dynamically, therefore no enumeration of contextual conditions is possible beforehand. The author explains that “Context arises from the activity. Context isn’t just there, but is actively produced, maintained and enacted in the course of the activity at hand”. In our work, as in the vast majority of CARSes, the representational view is adopted. Contextual conditions are enumerated beforehand by a system architect or expert of the domain. This provides a much simpler and computationally feasible approach, compared to the interactional view. Moreover, CARSs do not strive for a perfect model of the context (if such is possible at all). On the contrary, the main focus is on accuracy and explainability of the models,

sometimes preferring simpler yet computationally cheaper and precise model.

We call applications that take into account contextual information “context-aware” systems [Schilit and Theimer, 1994]. A good example of one classical context-aware system is the Conference Assistant [Dey et al., 2001]. This is a mobile context-aware device that supports users in several tasks. It lists the presentations that are relevant for a user, it keeps track of the user’s colleagues in order to share relevant information, and it helps to take notes about interesting presentations. At the conference registration desk the user receives a handheld device that runs the Conference Assistant application. When he starts the application, this automatically displays a copy of the conference schedule, showing the multiple tracks of the conference. When the user enters a room, the Conference Assistant automatically displays the name of the presenter and the title of the presentation. Context-aware computing is a new research field in computer science. It offers theoretical analysis on how to incorporate contextual information into software systems [Schilit and Theimer, 1994, Dey, 2001]. Early works on context-aware computing concentrated on motivating and explaining what context is and the technical aspects related to how to collect and store contextual information. In fact, Dey gives a very broad definition which should be refined in a concrete RS scenario.

In recommender systems context is usually playing the role of an additional information (besides users and items) which may be relevant at the current time to make a recommendation. There are several works that concentrate on the representational part of context. Ontology-based context-aware recommendation systems [Yu et al., 2006b, Sungrim Kim, 2007] use ontologies to represent semantics of the recommender knowledge. Representing concepts through ontologies can bring several advantages, such as enriching information when it is imprecise or incomplete, moreover, can support interoperability and the exchange of information between systems [Buriano et al., 2006]. For example, [Sungrim Kim, 2007] uses four types of ontologies: product, location, record and customer. The proposed method extracts a consumer’s preferred items based on his shopping history and recommends similar items according to the ontology. In fact, the algorithm can recommend not only the product, but also a group of products. A slightly simpler approach was proposed in [Adomavicius et al., 2005] where the authors use a hierarchical representation of contextual dimensions. In this work each dimension of the context has an associated hierarchy, which could be used to aggregate underlying data. Probably the most simple and widely used approach is the enumeration of variables, one for each contextual dimension [Domingues et al., 2009, Baltrunas et al., 2010a, Oku et al., 2006].

2.2 Recommender Systems

Recommender Systems collect and exploit various types of information about users, products, and interactions between users and products to generate a personalized list of items that fits user's current needs. This information may be gathered while the user is interacting with the system or can be acquired from other systems the user has interacted with. The most commonly used information about the interaction between users and products is the set of subjective ratings assigned by the users to previously experienced products. The system then uses these ratings for predicting the ratings for products not yet experienced. The products with the highest estimated ratings are then recommended to a user.

Recommender systems techniques can be classified into four main categories [Burke, 2002]: Collaborative Filtering, Content Based Filtering, Knowledge Based systems and Hybrid systems.

Collaborative Filtering (CF) is a recommendation technique which emulates a simple and effective social strategy called "word-of-mouth" [Hill et al., 1995, Shardanand and Maes, 1995]. Here, personalized recommendations for a target user are generated using opinions of users having similar tastes to those of the target user [Resnick et al., 1994]. This is a domain independent approach, which computes recommendations by leveraging historical log data of users' online behavior. Usually for this type of recommendations, the user must provide explicit opinion about an experienced item.

CF went a long way since Resnick et al. [Resnick et al., 1994] introduced CF to the field of news recommendations. An extensive survey on CF can be found in [Adomavicius and Tuzhilin, 2005] and [Schafer et al., 2007]. Resnick et al. originally concentrated their work on GroupLens. This system collects ratings in a distributed way and predicts the ratings of a target user by averaging the opinions of the most similar users. This idea was later extended by using more sophisticated nearest neighbors search algorithms and prediction rules [Koren, 2008]. Nowadays, Collaborative Filtering techniques are broadly classified into the *memory based* and *model based* techniques [Adomavicius and Tuzhilin, 2005]. Memory based CF falls into the class of Machine Learning techniques called lazy learners (or k-nearest neighborhood). Pure memory based methods require no computation at model building time, as they provide rating prediction based on the ratings of the closest neighbors - computed at the request time. Conversely, model based techniques use a training data set to build a predictive model that is later used to generate predictions. During the last five years the mainstream research in the area has shifted from memory based techniques towards model based techniques. This is mainly due to the Netflix prize competition [Bennett et al., 2007], that brought advanced Machine Learning and optimization techniques into the field of RS. In fact, it was eventually shown that model based approaches give higher

accuracy [Koren, 2008, Koren and Bell, 2010] than memory based methods.

Nowadays CF is considered among the most successful and widely applied RS techniques. The main motivation is that it requires no prior knowledge of the application domain, and it has been shown that when enough ratings are available the CF outperforms other techniques. However, CF suffers from the sparsity problem [Schein et al., 2002]; i.e., it fails to generate accurate recommendation when the available historical information about a user or an item is not sufficient. Moreover, it cannot adapt the predictions to the short-time preferences of the user [Hayes and Cunningham, 2004]. CF assumes that the ratings given in the past by a population of users for items can help in predicting the missing ratings, i.e., in some sense to predict the preferences of the users in the future. This assumption is valid only to some extent. In fact, the user’s interests can be relatively stable, and for many applications it is enough to consider the general user opinions about a set of items. However, the exact evaluation of an item, and therefore the appropriateness of a recommendation for that item, can be influenced by many additional and varying factors (i.e., context) that are not considered in CF. This thesis analyzes this issue and proposes extensions to CF in order to include contextual information into the recommendation process. An extended review of related work on CACF is provided in Subsection 2.3.

Content Based (CB) Filtering is another major technique that generates a recommendation for a user analyzing only the ratings of the *target* user. Content Based techniques analyze items’ descriptions to identify items that are of a particular interest to the target user [Pazzani and Billsus, 2007]. Typically, these CB systems learn the user profiles while analyzing the features of the items that were rated by the user in the past. Various machine learning techniques are exploited in order to learn these predictive models. Content Based Filtering plays a very specific role in RS literature, as it does not suffer from the new-item problem, i.e., the impossibility to recommend an item for which no rating was previously collected. Therefore, it is typically preferred to CF in those domains where the main need is to recommend recent items or data sparsity is very high [Pazzani and Billsus, 2007]. For example, in the news domain, a user should be informed also about articles that have not yet been rated by other users, if they are relevant for the user. The main down side of such systems is firstly a lower accuracy compared with CF, when more ratings are available. In addition, CB systems do not show serendipity [McNee et al., 2006], i.e., the user is usually recommended with “unsurprising” items containing similar content to those that the user evaluated in the past as interesting.

There are several approaches for generating content-based context-aware recommendations. Since content-based recommender systems are not our main focus, we will illustrate only a couple of important works in this field. Ghim-Eng Yap et al. in [Yap et al., 2005] employs Rank SVN [Joachims, 2002] to dynamically

understand which context is important for which user in the restaurant domain. Note that authors consider the features of an item (such as the restaurant category or the average price of a meal) as contextual features. A ranked list of 3 recommendations is given to a user and he is asked to select a single restaurant. The authors assume that the user's choice is influenced by the current context. The predictive model is built using information of the user responses. For example, if in context *C* the user has chosen the restaurant listed in the second position, it means that it is preferred to the first and the third restaurant. Such pair-wise preferences are used to learn the correct ranking of the restaurants for a given context. In this approach only the contextual aspects and not the textual description of a restaurant are used for generating recommendations. The trained system performed significantly better than the untrained system on simulated and real user evaluations. The authors discovered that, in choosing the restaurants, the users tend to consider only some aspects (payment available by cash, average price, and restaurant category).

Extending the previously quoted work, [Yap and Pang, 2007] proposes a Bayesian network based recommender that uses only a minimal set of context parameters to generate a recommendation. An interactive learning approach starts with the contextual information available for a specific user and uses cross-validation to identify the context that led to a good classification. In such a way only the most important contextual parameters are used in the final recommendation. The authors compared their method to a Decision Tree classifier (J4.8) with cleaned and uncleaned data and showed some improvement. In both papers the authors use features of the user and the items together with contextual features. Therefore, it is not clear how much the contextual information contributes to the prediction accuracy compared to the pure content based prediction methods. In this paper the authors also claimed that contextual information is very noisy, hence robust methods or data cleaning should be used.

Knowledge-based (KB) systems use a decision model to make inferences about the user needs. Usually a KB system computes or simply stores some recommendation rules designed by a domain expert [Schafer et al., 1999]. It can also use previous user interactions with the system and later exploit similar observed cases to generate a prediction [Lops et al., 2010]. The main advantage of KB systems is their flexibility to incorporate various types of rule, e.g., business rules, into the recommendation process. Moreover, these systems have no cold-start problem, as no user input is required to bootstrap the system. However, such systems are very expensive to build as they require accurate input from a domain expert. Because of that, pure knowledge based systems are not very common nowadays and usually they are combined with the previously mentioned techniques [Anderson et al., 2003]. Knowledge based system could be easily extended to use contextual information. An example of KB context-aware system

can be found in [Carolis et al., 2009], where the authors used common sense rules about weather, location, season and opening hours to filter out recommendations. The problem with such approaches is the exact quantification of how much a contextual feature influences the user preferences for an item.

Hybrid Systems combines several RS techniques to overcome the limitations of pure techniques [Burke, 2002]. Hybrid systems often provide better accuracy than pure techniques [Schlar et al., 2009]. Moreover, CF does not provide a direct way to integrate context into the prediction model. Therefore, a natural way to integrate such additional information is to use a hybrid system. In fact, most contextual post-filtering approaches would fall into the group of cascading hybrid methods [Anderson et al., 2003, Hayes and Cunningham, 2004]. Here rating prediction is done by CF and adapted by different predictive system that uses context. The classical pre-filtering CACF method [Adomavicius et al., 2005] can be also seen as a hybrid system. In fact, for each prediction made, the authors proposed to discard the ratings collected in the contexts that are different from the target one. The standard CF prediction method is used on this reduced set of data. Therefore, the reduction based approach could be seen as a type of a hybrid system where ratings are pre-filtered by one system and consequently used by the second system.

2.3 Historical Perspective of Context Aware Collaborative Filtering

This section gives a historical perspective on CACF. As we mentioned above, our main focus is Collaborative Filtering and therefore in this Section we will not survey other types of CARS. We will summarize the related work, and in particular we will try to point out how the quoted methods define the concept of context and how the context is actually used to generate recommendations. Throughout this Section we will illustrate how the research gradually shifted from memory-based models to model-based approaches. As we discussed before, context is a multi-faced term, therefore it is hard to give an extensive review of the CARS literature. To solve this issue, we will survey only those works that claim to introduce context-aware systems. Another extensive survey of CACF can be found in [Adomavicius and Tuzhilin, 2010]. It concentrates on some specific techniques and also lists definitions of context in various other fields.

2001 The research on Context Aware Collaborative Filtering started in 2001 with a technique that adapted the recommendation list to the specific task of the user [Herlocker and Konstan, 2001]. The interesting and strong point of this work is that these tasks could be specified without any metadata, content or

keywords. Instead, the user was asked to supply examples of items, related to the task at hand. For example, if the user is actually looking for an action movie recommendation, he would probably add action movies to his task profile. This would allow the system to create custom tasks, that could be hardly extracted using metadata. The task profile could be also extracted by the user behavior analysis, which is effortless for the user, but could be less accurate. The authors reported that 88% of participants accepted the context-aware version of the system, even if they needed to do some additional work for specifying their task profiles. In some sense this approach took an interactional view of context [Dourish, 2004]. The user self defines what is a specific context he wants to get recommendations for. According to Dey’s view of context, here a set of items characterized the current situation of a user. The major problem with this approach is the complexity of the task representation and the interpretation of the results.

Almost in parallel, Adomavicius et al. started the work on On-Line Analytical Processing (OLAP [Han et al., 2006]) like data representation for CF [Adomavicius and Tuzhilin, 2001]. The authors proposed a *multi-dimensional model* to store additional contextual information together with user ratings. The model extended the classical two-dimensional $user \times item$ matrix. Here, each rating is associated with additional information, denominated as “context” [Adomavicius et al., 2005]. This is a very general and expressive data model, which can be decoupled from the recommendation method itself. The same data model was used over and over again in various subsequent works by several authors [Adomavicius et al., 2005, Baltrunas and Ricci, 2009a, Karatzoglou et al., 2010]. This well defined data model can be viewed as a “representational” context-aware computing approach [Dourish, 2004]. Here the contextual dimensions, all the possible values and hierarchies, are defined explicitly. For example, to define the weather we could enumerate all the relevant values: sunny, cloudy, raining, snowing. This approach assumes that a domain expert or a system designer can a priori identify relevant contextual conditions. This in some way limits the user possibility to define his own context; however, it makes the selection (and automatic selection) of the target context model easier. Compared to the [Herlocker and Konstan, 2001] approach, here the user could only select a task from a pre-defined list, for example, “I want *a movie for my kids*”. According to Dey’s view, the context here is represented by a set of dynamic features, that describes an entity. The entity is the user, the movie and the experience of watching the movie.

2002-2004 In years 2004-2004 we saw several new applications for CARS [Hayes and Cunningham, 2004, van Setten et al., 2004]. See Section 2.4 for details on [van Setten et al., 2004]. One way to integrate the contextual information into CF recommender systems is to use hybridization techniques [Burke, 2002].

At first, the rating prediction is made using CF, and later it is adapted using context. Good example of such an approach was provided by Smart Radio [Hayes and Cunningham, 2004] - a web based music recommender system, where the user compiled and shared music playlists with the help of a CF system. It implements a “Cascading” hybridization approach, where the CF produces a candidate set that is then refined using contextual data. Genre and artist of the current song play the role of context, which is used to find the most similar cases from a candidate set. The authors reported a huge boost in performance when using this type of contextual information. Here the entity that context describes is the user who listens to a current track.

2005 The work on the multi-dimensional data model was later extended to a *reduction based CACF* [Adomavicius et al., 2005]. In this approach, the rating prediction is performed using the reduction based method. According to this method, firstly, the target user context is identified and the ratings collected in different conditions are discarded. Next, a standard CF prediction procedure is applied on this reduced set of data. Unfortunately, the experiments showed only minimal improvements in the recall. The authors also stressed the increase of data sparsity produced by this method and the lack of available contextually enriched data. The reduction based approach is used only in those data segments where it produces a significant improvement over the baseline method (tested using cross-validation techniques). The identification of these segments is also very expensive, but it could be done offline. As mentioned above even a careful selection of the best segments gave small advantages over the baseline CF.

At the same time a different approach was studied by Chen [Chen, 2005]. The author extended user-to-user CF by incorporating contextual information into the similarity computation step. First, the author proposed to use Pearson Correlation Coefficient to compute the similarity of two contexts based on the ratings that were given in these contexts. This similarity is later used to determine the rating context - the more similar is the context to the target context, the more influential is the rating. The final rating prediction combined all the weighted ratings, with respect to similarity in context, of all the neighbors, which is then further weighted with respect to the similarity of the user. The same idea was brought back in [Adomavicius and Tuzhilin, 2010]; however, to our best knowledge it was never empirically evaluated.

2006 All the previously described works used memory based approaches to incorporate context in CF prediction process. In 2006 Oku et al. [Oku et al., 2006] proposed to use a model based approach to compute user to user similarity. The authors used a memory based approach to compute the final recommendation. This started a new research line where context is incorporated into CF using

model based approaches. The work was extended in [Oku et al., 2007]; however, authors still used the same hybrid method combining model based and memory based approaches. In particular, the authors used support vector machine (SVM) classification method to compute the similarity between users in a specific context. For each user a SVM that maximizes the separation between the sets of liked and disliked items in various contexts was trained. The learned parameters of SVM were then used to determine if two users were similar. In particular, two users are similar if the corresponding SVMs classifies liked and disliked items in a similar way. [Oku et al., 2006] empirically showed that context-aware SVM significantly outperformed non-contextual SVM-based recommendation algorithm in terms of predictive accuracy and user's satisfaction with recommendations in a restaurant recommender system.

2007 In 2007 Anand and Mobasher [Anand and Mobasher, 2006] brought back the interactional view [Dourish, 2004] to model the context in CARS. They made the assumption that the observed user behavior is induced by an underlying context, but that the *context itself is not necessarily observable*. In fact, they proposed to model the context with a stochastic process that could be one out of d possible states. The problem is that d itself is not known in advance. Moreover, the meaning of each state (context) is also unspecified. For instance, it could happen that these states are found in the data not because of the context, but because of the impact of content related features of the items or because of stable user preferences. Moreover, without explicitly specifying it, the authors also use a representational view of context to model the short term user profile. They explicitly model the short term profile as the user browsing behavior during the current session.

More recent work of Adomavicius et al. [Adomavicius and Kwon, 2007] discussed multi-criteria recommender systems. In this setting, a user is supposed to evaluate several aspects of an item and later on these ratings on multiple criteria are used to recommend new items. This model could be seen as a context aware CF recommender system, where the user evaluates the same item under different contextual conditions. In fact this approach in some sense is similar to the multi-dimensional approach, where every user can have a different rating for the same item in different contexts. However, in multi-criteria rating systems the user is supposed to rate an item always with respect to all the criteria. This assumption is not realistic in context-aware recommender systems as the user cannot experience the item in all the possible contexts.

2008 In the new research area of CARS various methods were studied independently. During the tutorial given at RecSys'08 [Pu et al., 2008] Adomavicius et al. [Adomavicius and Tuzhilin, 2008] presented an interesting classification of CARS

methods. This analysis was extended in [Adomavicius and Tuzhilin, 2010]. The authors proposed to classify all CARS methods into three groups: pre-filtering, post-filtering and contextual modeling. The classification considers the time at which the context is used. Pre-filtering describes a hybrid system, where context is used to pre-process the data before the standard CF method is used. Post-filtering uses context to re-rank or filter-out the output of CF algorithm. Contextual modeling directly uses the context to make a rating prediction. Clearly, only some algorithms strictly fall into one of these groups. However, such classification allowed a better organization and understanding of the emerging CARS methods.

2009 The classification of CARS algorithms that we mentioned above gave a clearer picture of CARS research. Subsequently, there were some attempts to empirically compare the different paradigms. In [Panniello et al., 2009] authors try to figure out if and when it is better to use pre-filtering or post-filtering. They concluded that there is no superior paradigm and the winning approach depends largely on the data itself.

Another strong contribution to Context Aware Collaborative Filtering was conducted by Koren [Koren, 2009]. In the best paper award winner of KDD'09 the author used the temporal domain to improve the prediction accuracy of a CF algorithm. Koren showed that user ratings have temporal patterns that could be exploited to improve the accuracy of the model based approach. This work, following [Oku et al., 2006], made a step towards the model based approach for context handling. It also showed that each additional information that is associated with rating is valuable and should be used to improve the underlying algorithms.

In a series of works [Baltrunas and Ricci, 2009a, Baltrunas and Ricci, 2009b] Baltrunas and Ricci introduced the item-splitting method. In this paper the authors investigated an alternative method to improve the reduction based approach: in terms of algorithm complexity and also in terms of the prediction accuracy. They proposed a dynamic algorithm to automatically discover and use only the relevant contextual dimensions. Method splits the item profile, i.e., the collection of ratings assigned to an item by all the users, into two virtual items, if there is a statistically significant difference between the users' evaluations of this item in different contextual conditions. The work also confirmed the intuition that the more influential is the context, the more beneficial is to use CACF. Moreover, it showed that item splitting approach performed very similar, or better, compared to the reduction based approach for all the data sets used.

The item-splitting method was presented in the first Workshop on Context-Aware Recommender Systems, which was held in conjunction with Recsys'09 conference. This dedicated workshop organized by Adomavicius and Ricci opened a new research ground for innovative ideas. For example, Domingues et al. [Domingues et al., 2009] presented an interesting idea, where contextual informa-

tion is injected into the rating table as virtual items. Authors empirically tested this method with two top-N recommender systems, an item-based collaborative filtering and association rules. The results showed that the method is able to take advantage of the context when it is informative. In [Baltrunas and Amatriain, 2009] the authors presented the challenging problem of using contextual information with implicit data. The problem is very important, as it is cheaper to collect implicit responses from the user and therefore much more of such data are available [Hu et al., 2008]. However, the results were not conclusive and more research in this area is needed.

2010 Recsys’10 was a notable success in terms of number of works that analyzed CARS. Firstly, there were two workshops dedicated to this research area: 2nd Workshop on Context-Aware Recommender Systems (CARS-2010) and Challenge on Context-aware Movie Recommendation. Hideki et al. [Asoh et al., 2010] presented an important analysis that showed the relation between user responses in virtual and true contextual conditions. Here a virtual condition corresponds to a rating given when the user imagines that a certain contextual condition holds and gives an evaluation of an item in this situation. The authors reported that the user decisions are more logical in virtual context. Moreover, they have *larger* impact on the ratings in the virtual context. This has an important implication for all the off-line evaluations of CARS. This shows that the data sets must be better collected recording users ratings when they are in real situations characterized by some context variables. In [Baltrunas et al., 2010a] authors tried to address this issue and proposed a different approach to bootstrap the CACF system. Instead of asking user to rate the items in virtual context, they asked to specify what is the best context to consume the item in.

CAMRA Challenge on CARS brought new ideas into the field. First of all we need to mention the work that won the challenge [Shi et al., 2010]. Authors extended the classical Matrix Factorization approach to incorporate additional information about movie mood tags. It was done by first computing movie-to-movie similarity based on the tags and then using this additional information as additional modality of the prediction model.

Recently Karatzoglou et al. presented a work on the tensor factorization approach to CACF [Karatzoglou et al., 2010]. The model falls into the contextual modeling algorithmic paradigm. It describes a pure model-based technique to incorporate more than a single contextual condition into a CF prediction method. The model is very intuitive and extends the classical two-dimensional matrix factorization into an n -dimensional space, i.e., tensor factorization. The multi-dimensional cube model, which we described earlier [Adomavicius and Tuzhilin, 2001], is factored into lower-dimensional representation. Here the user, the item and each contextual dimension are represented with a lower dimensional feature vector. The authors compared the work with

two pre-filtering [Adomavicius et al., 2005, Baltrunas and Ricci, 2009a] memory based approaches and showed the substantial gain in accuracy for all the tested data sets.

2.4 Context-Aware Mobile Tourist Guides

Context-aware mobile tourist guides play an important role in CARS. In fact, they represent one of the major applications of these techniques. Most of the motivating examples quoted in CARS papers are taken from the tourism domain. A more detailed study of Mobile RSs can be found in [Ricci, 2010], and a survey of context-aware mobile computing can be found in [Chen and Kotz, 2000].

Mobile phones are becoming primary platforms for on-line information access [Ricci, 2010]. They enable users to access information services wherever they are, and especially on-the-move. This leads to more involving, always on-line types of communication. Information services are often consumed while traveling, e.g., detailed descriptions of Points of Interest can be found on-line. However, portable devices usually have a small (approximately 3.5-inch) display and they offer limited input capabilities. Most of the devices have a 12-key numeric keypad, or on-screen keyboard that makes communication (i.e., keyword based search) with a generic on-line systems challenging. The limited characteristics of these devices make information overload [Maes, 1994] even more critical and RS technology can play an important role to overcome these limitations. For this reason, the tourism domain has been a primary application area for mobile applications. Within the tourism domain Tourist Guides received the largest attention [Ricci, 2010]. Tourist guides usually support travelers in planning, organizing and executing a journey, i.e., helping to find relevant attractions and services, or supporting the exploration of an area. These systems usually recommend Places Of Interest (POI) (e.g., museums, art galleries, churches, etc.) as their primary type of items [Dunlop et al., 2004]. Moreover, current mobile devices are equipped with sensors that enable a context-aware information access. Thus, mobile RSs can largely benefit from the exploitation of information relative to the users' and items current context [Abowd et al., 1997, Chen and Kotz, 2000, Dey, 2001, Dourish, 2004].

The work on the Cyberguide [Abowd et al., 1997] opened the research on mobile tourist guides and gave initial contributions to the design of these systems. The proposed model implements spatial awareness of the mobile device, history tracking, but it lacks the recommendation function.

A more sophisticated mobile context-aware city guide is COMPASS application [van Setten et al., 2004]. The authors use context both as a soft and hard criterion for recommendations. Hard criteria filter out irrelevant recommendations, and soft criteria modify the final recommendation list for a user. This work concentrates more on the system architecture and the user study, evaluating

the usefulness of a context-aware guide, whereas the recommendation step is described very briefly. The authors propose different prediction strategies for different classes of POIs. As the types of POIs are described by an ontology, the recommendation engine is aware of the class hierarchy of each POI. The user can browse a map indicating her current location and a selection of nearby buildings, buddies, and other relevant objects determined by considering her user profile. The map and the objects shown are updated when the user moves (context changes) or when the user profile or the goal change. The authors conducted a user study that showed several interesting results: a) in general most users evaluate context-awareness (time and location) as useful; b) half of the users do not like “last time visited” feature, which lowers the predicted relevance of the restaurant if the user yesterday has visited a similar restaurant; c) a lot of people think that “application becomes too intelligent”. It is worth noting that in this system the user can deliberately specify the contextual conditions that matters.

[Ahn et al., 2006] presents an approach to mobile context-dependent recommendations that extends the classical collaborative filtering (CF) method by using information about the user and the item location, the time of the recommendation and the type of the user needs: either hedonic, or neutral or utilitarian. The recommendation process starts by collecting the user position, the time and the needs and filtering out the items that are not located close to the user position. Then, in order to apply CF, it searches for similar users, using a particular similarity metric. This metrics combines (by multiplication) the standard adjusted cosine metric [Sarwar et al., 2001] between the active user and a neighbor user, and a measure of the similarity of the current time, position and needs between the two users. The authors collected their own rating data about shopping places in Korea and compared several CF algorithms with their proposed model showing a slightly better performance (for mean absolute error [Herlocker et al., 2004]) of their approach. The idea of using the location of the user to tune the user-to-user similarity function has also been exploited by [Horozov et al., 2006]. In their restaurant recommender system (Geowhiz) they assume that people who live in the same neighborhood are likely to visit the same nearby places. Hence, since people can be correlated in CF only if they have co-rated items, they infer that there is a higher probability of correlating people who live close to each other than correlating people who live further apart.

In addition to adapting to the more classical categories of user preference and situation context (e.g., location and time) [Yu et al., 2006b] introduces the “capability context”, i.e., device and network capability, as an input for both content and presentation recommendations. In order to deal with all the three context categories, they use a hybrid recommendation approach exploiting content-based methods, a Bayesian classifier, and rule-based methods. Their context-aware media recommendation platform is called CoMeR and supports media recommendation, adaptation, and delivery for smart phones. It is worth noting that they

used an ontology-based context model for context representation. This model adopts OWL as a representation language to enable expressive context description and data interoperability with third-party services and applications. They also adopt the multidimensional model proposed by [Adomavicius et al., 2005], so for instance their recommendation output is not limited to a standard user-adapted rating prediction, but it can generate a more specific rating prediction for each type of user mobile device or type of output for the same item recommendation (e.g., image vs. text description of the item).

Part II

Algorithms for CACF

Item Splitting

In this chapter we present a CACF approach that we have called *item splitting*. It enhances CF by taking into account contextual conditions. Using contextual information the system could fine tune the recommendations not only to the user preferences (ratings), but also to a particular contextual condition. Here we have enriched the standard 2-dimensional CF matrix with a model of the context comprising a set of features either of the user, or the item, or the evaluation. In item splitting the set of the ratings for an item are split into two subsets according to the value of a contextual variable, e.g., ratings collected in “winter” vs. those collected in “summer” (here the contextual variable is the season of the rating/evaluation). These two sets of ratings are then assigned to two new fictitious items (e.g. beach in winter and in summer). This split is performed only if there is statistical evidence that under these two contextual conditions the item’s ratings were different, i.e., users evaluate the item differently.

Item splitting was firstly introduced in [Baltrunas and Ricci, 2009a] and [Baltrunas and Ricci, 2009b]. [Baltrunas and Ricci, 2009a] defined the approach and mainly investigated some variations of the algorithm that are related to the splitting criteria. In [Baltrunas and Ricci, 2009b] we compared item splitting with the classical context aware CF approach proposed by Adomavicius et. al [Adomavicius et al., 2005], which is called Reduction Based. In this chapter we provide an extensive and comprehensive evaluation of our proposed approach. We have performed some new experiments using a real world data set that has been also used by [Adomavicius et al., 2005]. This enable us to better compare item splitting with the state of the art algorithms. Moreover, we illustrate how one can deal with missing contextual information and we evaluate the proposed approach that is tailored to item splitting. In addition, we extended the evaluation considering other performance metrics such as precision/recall, in addition to the prediction accuracy (MAE), and we measured how the contextual conditions

can modify the top-k recommendations when the context changes.

This study shows that standard neighborhood and matrix factorization based CF models cannot cope with rating data influenced by contextual conditions. In fact, we show that if the contextual condition does influence the item ratings, then item splitting techniques can help to improve the accuracy of CF, especially with matrix factorization techniques.

The closest approach to item-splitting is the classical Reduction Based context-aware approach by Adomavicius et al. [Adomavicius et al., 2005]. Both methods use data pre-processing in order to modify the training data. However, they have noticeable differences. First of all the Reduction Based approach uses a “wrapper like” [Kohavi and John, 1997] method to compute the best contextual segmentation. Accuracy of the underlying prediction method is tested while using each of the contextual segments. This algorithm searches for the contextual segments where a better rating prediction is obtained by considering only the data contained in the segment rather than using the full data set. Then, the final prediction for a target user is done using only the data in the segment containing the target context only if this has a higher prediction accuracy than the model built using all the data (irrespectively from the context).

This is a very expensive approach, as there is an exponentially growing number of segments depending on the number of contextual dimensions, and for each of them one should build and test a predictive model. Moreover, to improve the reliability of the segment accuracy estimations, the authors proposed to use cross-validation. Instead, item splitting uses a “filter like” method [Kohavi and John, 1997], where each item is tested for a split using a simple measure of fit such as Information Gain or chi square statistic. Another difference is that item splitting considers to split each item separately. This is a more dynamic approach that could be suitable in the application areas where only some of the items’ evaluations depend on the contextual conditions. However, one should take care of not over fitting the data as item splitting is a more complex method compared to Reduction based. The last difference is that item splitting uses all the ratings to make a prediction in any context, whereas Reduction based uses only the ratings collected in the same context as that of the target user.

The rest of the chapter is structured as follows. Section 3.1 describes the details of our approach and provides the complexity analysis of the proposed algorithm. Section 3.2 illustrates the experimental evaluation of the approach. It starts with the experimental setup and provides the results of a first set of experiments where we used a small, real world, context-tagged data set of ratings. Then we analyze the performance of the proposed algorithm on a bigger data set that uses user demographic data as context and a number of semi-synthetical data sets, that allowed us to study the various properties of the algorithm. Finally Section 3.3 draws the conclusions and shortly lists our future work.

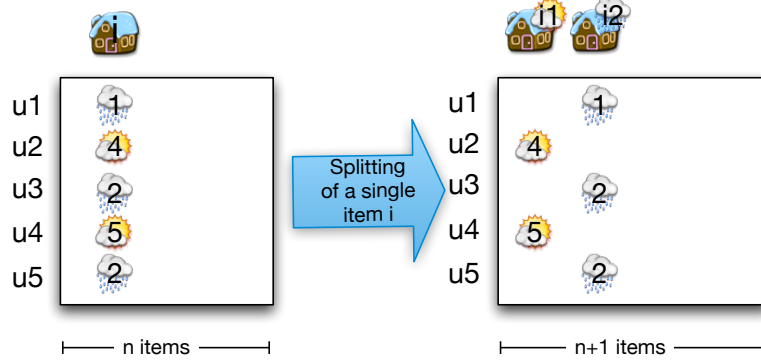


Figure 3.1: Item splitting

3.1 Method

Our rating model extends the traditional CF data model by assuming that each rating r_{ui} in a $m \times n$ users-items matrix, is stored (tagged) together with some contextual information $c(u, i) = (c_1, \dots, c_k), c_j \in C_j$, describing the conditions under which the user experience was collected. Here, c_j is a nominal variable which is drawn from a set of possible values for the contextual dimension C_j . For example, $C_{weather} = \{sunny, cloudy, raining, snowing\}$. The proposed method identifies items having significant differences in the ratings tagged with different contextual conditions (see later the exact test criteria). For each one of these items, our algorithm splits its ratings into two subsets, creating two new artificial items with ratings belonging to these two subsets. The split is determined by the value of one contextual variable c_j , i.e., all the ratings in a subset have been acquired in a context where the contextual feature c_j took a certain value. So, for each item the algorithm seeks for a contextual feature c_j that can be used to split the item. Then it checks if the two subsets of ratings have some (statistical significant) difference, e.g., in the mean. If this is the case, the split is done and the original item in the ratings matrix is replaced by the two newly generated items. In the testing phase, the rating predictions for the split item are computed for one of the newly generated items. For example, assume that an item i has generated two new items i_1 and i_2 , where i_1 contains ratings for item i acquired in the contextual condition $c_j = v$, and i_2 the ratings acquired in context $c_j \neq v$, hence the two sets partition the original set of ratings.

Figure 3.1 illustrates the splitting of one item. As input, the item splitting step takes a $m \times n$ rating matrix of m users and n items and outputs a $m \times (n+1)$ matrix. The total number of ratings in the matrix does not change, but a new item is created. This step can be repeated for all the items having a significant

dependency of their ratings on the value of one contextual variable. We focus on a simple application of this method where an item is split only into two items, using only one selected contextual variable. A more aggressive split of an item into several items, using a combination of features, could produce even more “specialized” items, but potentially increasing data sparsity. We note again, that a user can in principle rate the same item in several contexts. Therefore, the ratings for items i_1 and i_2 could overlap, i.e., there could be users that have rated i twice but in different contextual conditions.

So far we explained how item splitting uses contextually tagged ratings’ data in the pre-processing step. Then, to build the recommendation list for a user we use a standard CF approach [Adomavicius and Tuzhilin, 2005]. Namely, first we generate rating predictions for all the items unexperienced yet by a user, then we sort them according to the predicted rating value, and finally we make recommendations as the top- k items with the highest predicted ratings. However, in our case the recommendation list also depends on the context. Therefore, assume that the system needs to compute a rating prediction for the item i and user u in a context where $c_j = x$. Then the prediction is computed for the item i_1 if $x = v$, or i_2 if $x \neq v$, and is returned as the prediction for i . Note that the prediction will depend not only on the item i and the user u but also on the contextual feature $c \in C$ that was previously used to split the item i . To build context-aware recommendations we assume that we know the current (target) context of a user (e.g., is watching a movie with a companion), of the item (e.g., there is a discount for it) and of the situation (e.g., the current weather is “sunny”). We also notice that the system is able to build predictions for all the combinations of $user \times item \times context$. As a matter of fact, building a single recommendation list for a user could involve computing rating predictions using several contexts, as context itself depends not only on the user, but also on the item or the combination of both. For example, distance from user to the item would change for each $user \times item$ pair. Therefore, for each rating prediction for the same user we could use different context C . Clearly, some contextual conditions depend only on the user (such as current user task) and therefore is the same for all the target predictions. For instance, in Section 3.2.4 we discuss the results of an experiment, where we considered and studied the effects of the changes in the values of a contextual feature that remains constant for all the rating predictions of a given user.

3.1.1 Splitting Criteria

We conjecture that the splitting of an item could be beneficial if the ratings of that item in each of the two alternative contextual conditions are more homogenous than the original set of the ratings for the item. Alternatively, splitting could be useful if these two new sets of ratings have different properties, i.e., for instance, if

the average of the ratings for an item in context $c_j = v$ is significantly larger than the average of the ratings of the same item when $c_j \neq v$. One way to accomplish this task is to define an impurity criteria t [Breiman et al., 1984], which depends on the item i that is being split and the actual split s . A split s is determined by selecting a contextual variable and a partition of its values in two sets. Thus, the space of all possible splits of item i is defined by the context model C . So, if there are some candidate splits $s \in S$, which divide the ratings for i into two sets, we choose the split s that maximizes $t(i, s)$ over all possible splits in S .

We considered five impurity criteria: t_{mean} , t_{prop} , t_{IG} , t_{chi} and t_{random} .

- $t_{mean}(i, s)$ impurity criteria is defined using the two-sample t-test and computes how different are the means of the ratings in the two rating subsets, when the split s is used. The bigger the t value of the test, the more significant the difference of the means in the two partitions is:

$$t_{mean} = \left| \frac{\mu_{i1} - \mu_{i2}}{\sqrt{s_{i1}/n_{i1} + s_{i2}/n_{i2}}} \right|$$

where μ_i is the mean rating of the item i , s_i is the rating variance of item i and n_i is the number of ratings that item i contains.

- $t_{prop}(i, s)$ uses the two-proportion z-test and determines whether there is a significant difference between the proportions of high and low ratings in i_1 and i_2 , when s is used. We consider two rating classes. A rating is high if its value is 4 or 5, and low if it is 1, 2 or 3 [Herlocker et al., 2004]. For data set with the rating scale from 1 to 13 we consider rating value to be high if it is more or equal to 8. To test the difference between proportions we use the two-proportion z-test computed as:

$$t_{prop} = \frac{p_{i1} - p_{i2}}{\sqrt{p(1-p)(1/n_{i1} + 1/n_{i2})}}$$

where $p = (p_{i1}n_{i1} + p_{i2}n_{i2})/(n_{i1} + n_{i2})$, p_{i1} (p_{i2}) is the proportion of high ratings in i_1 (i_2), and n_{i1} (n_{i2}) is the number of ratings in i_1 (i_2).

- $t_{IG}(i, s)$ measures the information gain (IG), also known as Kullback-Leibler divergence [Quinlan, 1993], given by s to the knowledge of the item i rating class (low or high):

$$t_{IG} = H(i) - H(i_1)P_{i1} - H(i_2)P_{i2}$$

Here $H(i)$ is the Shannon Entropy of the item i rating class distribution and P_{i1} (P_{i2}) is the proportion of ratings that i_1 (i_2) receives from item i .

- $t_{chi}(i, s)$ computes chi square test for proportions to determine if there is a significant difference between the proportions of high and low ratings in i_1 and i_2 . This criteria is similar to $t_{prop}(i, s)$, however, uses different proportion statistic.
- $t_{random}(i, s)$ is used as a reference for comparing the behavior of the other methods. It returns a random score for each split s .

3.1.2 Complexity of the Algorithm

The overall algorithm time complexity is exponential in the number of values that a contextual feature can take and could be expressed as $O(nm2^d)$, where m denotes the number of users, n is the number of items, and d is the maximum number of values a contextual feature can have. Clearly, the execution time depends linearly on the number of items n in the data set, the number of the ratings for each item and the number of possible splits for each item. In fact, the proposed algorithm first splits an item in all the possible splits and then it computes the statistic on each split. Given a split, our used statistics can be computed in a linear time with respect to the number of ratings, and the maximum number of ratings is m , i.e., the number of users. The algorithm splits the item only in two subsets, therefore, the number of possible splits using a single feature can not exceed $\frac{2^d-2}{2}$ splits, where d is the maximum number of different values of a single contextual feature. Here, $2^d - 2$ is the number of proper subsets of a set of cardinality d (i.e., excluding the empty set and the set itself). For each one of these subsets we have a partition, but we count two times the same partition since the same partition is created by a set and its complement. Also note that usually items are not rated by all the users. Moreover, some splits have not enough ratings to test a statistics and are therefore discarded from the candidate split list. Conversely, it is clear that the execution time could increase for data sets that have a large number of contextual features taking many different values.

In terms of execution time the item splitting algorithm performed very well. On two data sets that we have used ($d=5$, $n=192$, $m=84$ for the real world and $d=3$, $n=11K$, $m=7K$ for the semi synthetic data sets) item splitting procedure was executed in less than 10 seconds using a desktop machine with 2.2 GHz processor and python as the implementation language. In comparison, to train the Matrix Factorization (**MF**) prediction model (described later), which is implemented in C programming language, was required approximately one minute (for the bigger data set). Therefore, item splitting does not add a large overhead to the overall execution time when using data sets of similar dimensions.

3.1.3 Missing Context Values

Real world ratings' data sets tend to be incomplete, i.e., users rate a small minority of the items. Moreover, the collected ratings are usually noisy, i.e., the rating of a user for an item is typically the sum of a quantity measuring the true user satisfaction for the item plus an error term [Han et al., 2006]. The noisy and incomplete nature of contextual information is also due to the fact that the state of contextual features is usually automatically collected by low level sensors [Dey, 2001], or actively asked to the user [Adomavicius et al., 2005]. In many cases the user does not respond with an answer or the communication link to the sensor is not reliable, thus no or noisy information about the current contextual condition is obtained.

Our prediction model relies on contextual features and therefore, we need to elaborate a solution to generate recommendations even if some of the contextual information is missing. In data mining several approaches for dealing with missing values have been proposed [Han et al., 2006]. Such techniques are general and could be applied in the context-aware CF case. The most common approach is to ignore the tuple that contains missing data. However, in this way we would loose a lot of valuable information only because a single entry of context is missing. In fact, for some data sets most entries could have partially unknown contextual information. Another popular solution is to use the most common or the most likely value in place of the missing one. In this case we would risk to bias the prediction towards the most common context.

Instead of relying on these general approaches, we have designed a technique that is tailored to item splitting. As described earlier, when generating candidate items with respect to a split, we put a rating to one of the two newly introduced fictitious items. The allocation of a rating to one of the two new items depends on the value of the contextual attribute, on which the split was made. If we miss such an information, we propose to assign the rating to both items. Note, that if we would use the approach mentioned earlier, i.e., guessing the more likely value of the missing context variable, the rating would be assigned to only one of the two newly generated items.

Besides, when a rating prediction must be computed for a target user-item combination, there could be cases where we do not know the target context C , as some values of the contextual attributes could be missing. Hence, when making a rating prediction for an item i in context C , we first determine if the item i ratings were split by the algorithm into two items i_1 and i_2 and what contextual attribute $c \in C$ was used to perform the split $s \in S$. If the ratings of i were split using c , but we do not know the current value of this contextual attribute we make a rating prediction for i_1 and i_2 and return the final rating prediction as the average of the two predictions.

The evaluation of this approach for dealing with missing contextual values is

presented in Subsection 3.2.5.

3.2 Experimental Evaluation

This section illustrates the results of an experimental evaluation where we used two real world data sets and also some semi-synthetic data sets that we generated ad-hoc to test the behavior of the proposed context-dependent CF algorithm under controlled conditions. We start the Section by introducing our experimental setup. Subsection 3.2.1 compares the performance of different context-aware CFs with different splitting criteria on a real world data set. Subsection 3.2.2 uses another real world data set that is considerably bigger. However, here user demographic information is used to split items instead of a truly contextual condition. We evaluated the performance of different prediction methods while splitting an increasing amount of items. Then in the following Subsections we describe the usage of semi-synthetic data sets and we analyze various properties of our algorithm together with another context-aware CF method. Subsection 3.2.3 shows the performance of our method when the influence of the contextual feature on the rating value increases. Subsection 3.2.4 shows the amount of changes in top-k recommendation list, when context changes. Subsection 3.2.5 analyzes the performance of item splitting with an increasing number of missing contextual features. Subsection 3.2.6 concentrates on analyzing context free and two context-aware CF methods.

Datasets. We tested the proposed method on two real world and a number of semi-synthetic context-dependent data sets. Here we provide only a brief summary of the used data sets and full description can be found in Appendix C on page 155.

For the first experiment we used the data set provided by Adomavicius et al. [Adomavicius et al., 2005]. The data set contains 1464 ratings by 84 users for 192 movies (hence a rather small data set compared with other standard data set used in CF research). In our experiments we used 5 contextual features: a)companion {friends, parents, girlfriend, alone, colleagues}, b)day of the week {weekday, weekend, don't remember}, c)if it was opening weekend {yes, no, don't remember}, d)would the user recommend it to a friend {Excellent, Good, Fair, Bad}, e) year seen {2000,2001,2002}.

The second real world data set that we used was provided by [Yahoo, 2007] Yahoo! Webscope research project. It is much bigger than the data set described above and contains 221K movie ratings with scale {1, 2, 3, 4, 5}, for 11,915 movies by 7,642 users. The Yahoo! data set contains user age and gender features. We used 3 age groups: users below 18 (u18), between 18 and 50 (18to50), and above 50 (a50). The gender feature played the role of a contextual variable in

the item splitting algorithm. That is, we could split an item’s ratings, generating two fictitious items, one containing the ratings of the male and the other of the female users.

In order to control the influence of the contextual features and analyze various properties of the algorithm we also generated some semi synthetic data sets. These were generated using the original Yahoo! data set. We modified the original Yahoo! data set by replacing the gender feature with a new artificial feature $c \in \{0, 1\}$ that was assigned randomly to the value 1 or 0 for each rating. This feature c is representing a contextual condition that could affect the rating. Thus, in these data set the contextual condition is more “influencing” the rating value as α and β increase. See Appendix C for details.

Prediction methods. We computed the rating predictions with three techniques: user-based CF (KNN), matrix factorization (MF) and a non-personalized recommendation computed as the item-average (AVG). In KNN we used Pearson Correlation as user-to-user similarity metric and when making a rating prediction for a user we considered only the neighbors that have rated the target item and have co-rated a minimum of 6 items with the target user [Berkovsky et al., 2007]. Matrix factorization uses slightly modified gradient descent based matrix-factorization algorithm provided by Timely Development [Timeley, 2008]. We used a single validation set to find the best parameters for the two CF methods. KNN uses $k=30$ nearest neighbors for the Yahoo! and synthetic data sets, whereas, MF uses 60 factors and the other parameters are set to the same values optimized for the Netflix data set. It might not be the best setting, but all the system variants that we compared used the same settings.

Evaluation Measures. To evaluate the described methods we used 5-fold cross-validation and measured Mean Absolute Error (MAE), precision and recall [Herlocker et al., 2004]. MAE is frequently used in the CF literature, and represents the mean difference between the predicted and actual rating of users. The usage of precision and recall in recommender systems needs some clarifications. These measures can be only estimated since to compute them precisely one would require the knowledge of the rating (relevance) of each item and user combination [Herlocker et al., 2004]. Usually there are thousands of candidate items to recommend (11K in our case) and just for a small percentage of them we know the true user’s evaluation (typically less than 1%) . Herlocker et al. [Herlocker et al., 2004] proposed to estimate these measures by computing the prediction just for $user \times item$ pairs that are present in the ratings data set and consider items worth recommending (relevant items) only if the user rated them 4 or 5 when the rating scale is $\{1, 2, 3, 4, 5\}$ and 8 and greater if the scale is from 1 to 13. We computed the measures on full test set (of each fold), while

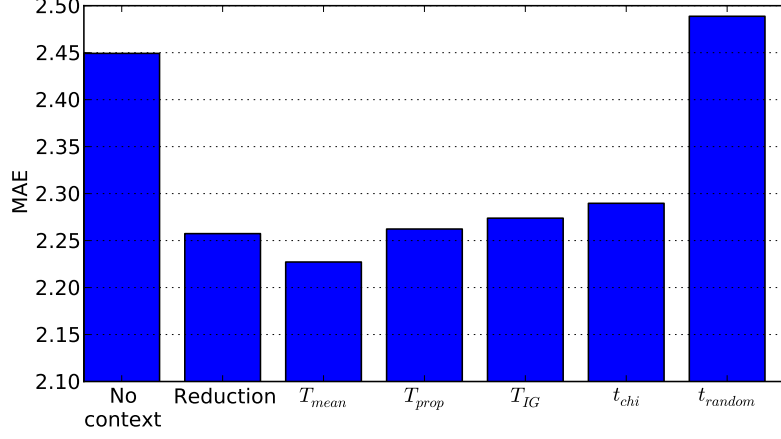


Figure 3.2: Comparison of contextual pre-filtering methods for real world data set.

trained the models on the train set.

3.2.1 Performance on Contextually-Tagged Data

The first experiment was conducted on data set provided by Adomavicius et al. [Adomavicius et al., 2005]. In the original experiment the authors used the *KNN* prediction method, and reported their results for precision, recall and F measure. Moreover, they used MAE for optimization purposes, i.e., for segment selection. In this work we have used the *MF* rating prediction method and we have used MAE both for testing the prediction and in optimization (training) step to build our model. We choose *MF* as it usually outperforms *KNN* [Koren, 2008] and is now considered the state of the art method for CF predictions [Koren et al., 2009].

Figure 3.2 compares the Reduction Based approach [Adomavicius et al., 2005] and items splitting with respect to a baseline method that does not take into account contextual information. Note that all the three methods ultimately use the same prediction method, namely *MF* and the only differences are in the pre-filtering step of the input data. Here, item splitting uses various impurity criteria discussed in Subsection 3.1.1 whereas Reduction Based determines the segments of the data where contextual information is useful to improve MAE. We see that both the context-aware CF systems have a better prediction accuracy compared with the context-free approach. The only exception is item splitting when it uses the random splitting criteria; but this was expected. In fact, when splitting 20% of items it performs worse than all the other methods. We further note that the best performing method is item splitting with T_{mean} as splitting criteria; it improves the baseline prediction by 9%. The second best performing

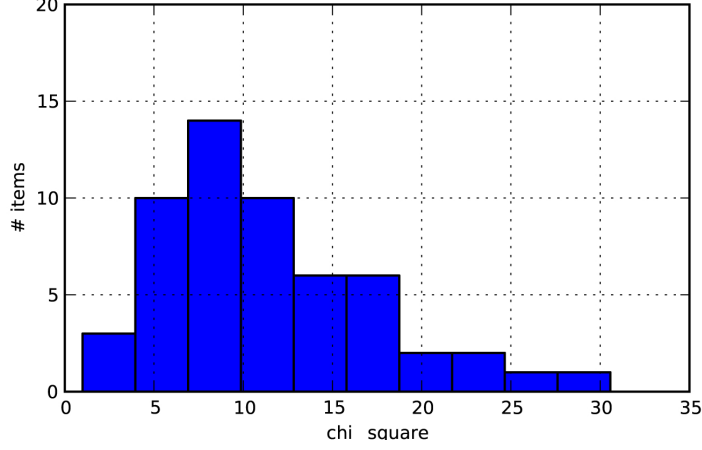


Figure 3.3: Distribution of chi square statistics for items with possible splits.

method is Reduction Based that improves the context free method by 7.8%. The improvement obtained with Reduction Based confirms the validity of this method that was originally introduced in [Adomavicius et al., 2005]. Other item splitting applications, i.e., with other impurity measures, out-perform the context-free method as well. However, they provided slightly smaller improvement: T_{prop} 7.6%, T_{IG} 7.16%, T_{chi} 6.18%. Note that the three best performing methods have very similar accuracy. The differences are small and the best method could depend on the careful parameter tuning.

For the T_{prop} , T_{mean} and T_{chi} methods we split the item if the test statistic was greater than 4. This threshold approximately corresponds to the 0.05 level of statistical significance. The exact values are: 3.84 for chi square with 1 degree of freedom, 4.03 for t-test. We set threshold of T_{IG} equal to 0.05 by trying several values. It is likely that one could better fine-tune the parameters to improve the performance.

Figure 3.3, shows the distribution of chi square statistics for each item in the training set. We assigned an item to a histogram bin by looking at the maximum value of the statistic. In other words, among all the possible splits of an item we report the one with the highest chi-square statistic. For example, imagine that the item i could be split using the splits s_1 and s_2 , and such splits would produce the chi square statistics 0.01 for s_1 and 30.1 for s_2 . In such case, we would add item i to the last bin of the histogram.

Note, that we did not split the items with few ratings, i.e., if the split would generate items with less than 5 ratings. There are only 58 items out of 192 that could be split. This number can be computed by summing up all the values of the histogram bins. For the remaining $192 - 58 = 132$ items in the data set there

	Male	Female	u18	18to50	a50
#ratings	158,507	52,224	45,084	157,844	7,803
mean	4.03	4.23	4.17	4.06	3.99

Table 3.1: Rating statistics for different demographic groups

are no such split that would result in two fictitious items containing at least 4 ratings each. This shows that item splitting can not perform in an optimal way on this data set, as many items do not have enough ratings.

3.2.2 Evaluation on Data containing Demographic Information

In a second experiment we analyzed the algorithm performance on the much bigger Yahoo! data set. Initially, we made a general statistical analysis of the data. We used the two-proportion z-test (as described above) and measured if the ratings of the two genders, or the ratings of different age groups, show statistically significant differences. For the Yahoo! data set the biggest statistical difference was obtained for the gender attribute (z-score 25.8), i.e., males and females do rate (on average) the items differently. For instance, females rate on average higher than males. The summary of the Yahoo! data statistics is showed in the Table 3.1.

The two-proportion z-test statistics shows that different demographic groups rate movies differently. However, the difference in the means of the ratings are small. Moreover, when user-to-user CF makes a rating prediction it scales the rating according to the user mean. Therefore, such differences in the means can be captured by the underlying CF algorithm. For example, *KNN* incorporates the average user rating into the prediction step of the algorithm. In fact, in this case we observed that Reduction Based could not find any segment that improves the prediction accuracy of the baseline approach. Therefore, the prediction accuracy of the Reduction Based approach for this data set is the same as any standard method without contextual pre-filtering.

Conversely, when using item splitting with t_{IG} criteria pre-filtering slightly improves the performance of the *MF* CF algorithm. When splitting 1% of the items with the highest impurity the MAE computed for the whole data set is decreased by 0.1%. The change is small, because most of the predictions in the test data set are not affected by pre-filtering since a small amount of items are split. However, for a small set of items, such as the romantic story “Chocolate” there is a significant difference in the average ratings of the men and women. The average male rating was 4.2 (60 ratings), and average female rating was 4.8 (83 ratings). When splitting more items we observed a decrease in the overall

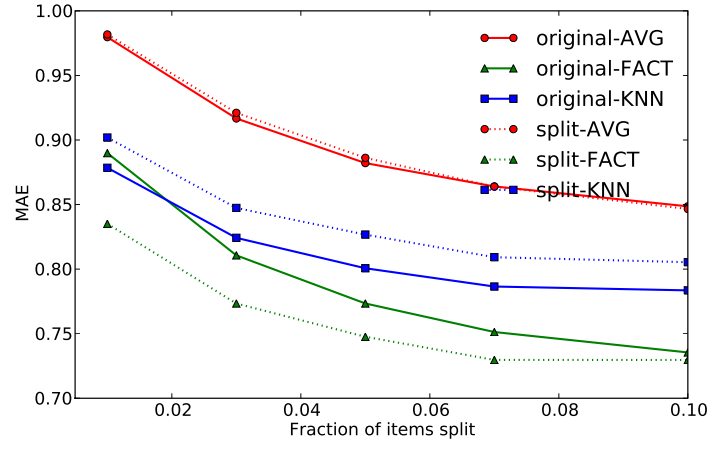
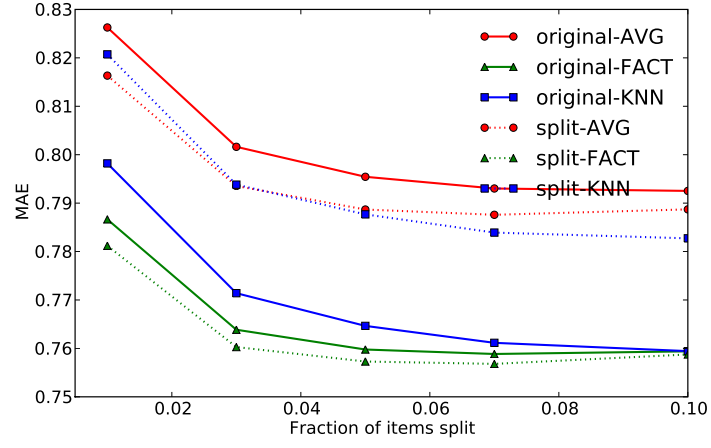
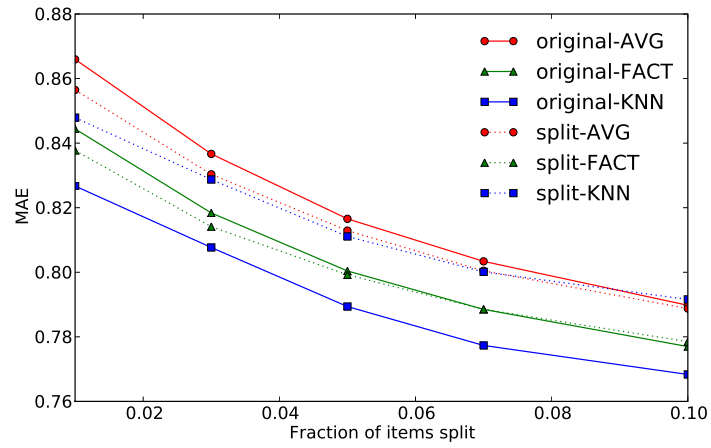
(a) t_{IG} (b) t_{prop} (c) t_{mean}

Figure 3.4: Item splitting for various criteria

performance. This can be explained by the fact that there is no strong functional dependency between gender, age and the rating. Therefore, splitting the items using a contextual feature that produces a rather arbitrary split (if there is no strong evidence to support the split) is not beneficial. Moreover, splitting items makes data more sparse and the computation of item-to-item correlation, which is used in *KNN*, could become unreliable. We also measured the performance of the other proposed split criteria on the full data set. MAE of *MF* increased: 0.3% for t_{prop} , 0.3% for t_{size} , 0.3% for t_{mean} , 0.05% for t_{random} .

In conclusion, item splitting has a small effect when applied to the gender and age features in the Yahoo! data set. Besides, the only splitting criteria that improved the accuracy in the full data set is t_{IG} ; it can improve the accuracy of the prediction for ratings belonging to split items (as it will be shown later) and also for the others. These improvements are small because, as said above, there is not a significant dependency between these contextual features and the rating behavior of the users. In fact, as we will show in the next section, when the dependency from the context is stronger then item splitting is more effective.

To emphasize the effect of item splitting using different split criteria, we computed MAE for the rating prediction only of the items that were actually split. We split an increasing percentage of the items, selecting those with the highest impurity. We compared the rating prediction accuracy using the split data set with that obtained for the same ratings using the original data set (not split). The results for various impurity criteria on the Yahoo! data set are shown in Figure 3.4. With the exception of t_{IG} , item splitting improves the performance of the non-personalized *AVG* method (original-AVG vs split-AVG in the figures). When 1% of the items (with highest impurity) are split the improvements are as follows: -0.2% for t_{IG} , 1.1% for t_{prop} , 1.0% for t_{mean} and 0.4% t_{random} . The improvements are small and this basically depends on the fact that gender and age do not significantly influence the prediction. We also observed that *KNN* was negatively affected by item splitting (both, t_{IG} and t_{prop}), and MAE increased (original-KNN vs split-KNN). We conjecture that this can be due to the reduction of the number of ratings in the target item profile when split is applied. We initially optimized the number of nearest neighbors (k parameter) to 30. But, after the split, the target item will have a smaller number of ratings (the average size of an item profile is 19 ratings) and *KNN* will tend to include all the users that have rated the target (without making any user selection). In fact, to avoid such an effect, we should optimize k for each data set separately (pre-processed and original). Conversely item splitting is strongly beneficial for *MF* when splitting 1% of the items with the highest impurity the improvements are as follows: 5.6% for t_{IG} , 0.4% for t_{prop} , 0.7% for t_{mean} , 0.9% for t_{random} . Here, notably the best performance is achieved by t_{IG} . t_{IG} measures the information brought by the contextual variable and is very different from all the other criteria used.

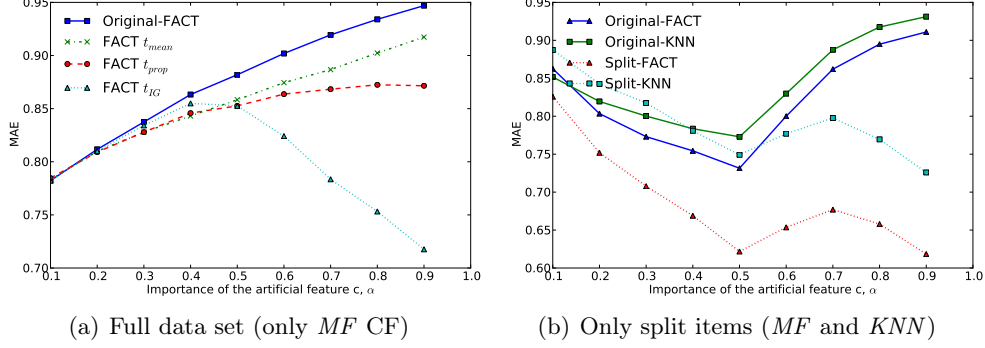


Figure 3.5: Effect of a contextual feature

3.2.3 Varying the Impact of the Contextual Variables

For better understanding the potential of item splitting we tested this approach on the semi-synthetical data sets described earlier. Here we used the data sets with fixed $\beta = 1.0$ and varying $\alpha \in [0.1, 0.9]$. That is, we varied the number of items that are affected by the contextual feature, but if an item is affected, then all its rating are increased or decreased according to the value of the contextual feature. The larger is the value of α the more influential is the contextual feature on the ratings. Figure 3.5(a) compares the MAE of *MF* for three different splitting criteria varying the impact of the context feature c on the ratings, as measured by the α parameter. Here, the different curves shown in the figure are obtained by splitting the items if the p-value of t_{mean} and t_{prop} tests are lower than 0.05, and when t_{IG} is greater than 0.18 (in the previous experiment these values produced the split of 5% of the items).

Figure 3.5(a) shows that in these context-dependent data sets, increasing the value of α , i.e., increasing the number of items whose ratings are modified according to the value of the context feature, the overall MAE increases. So this dependency of the ratings from a contextual condition plays the role of noise added to the data, even if this is clearly not noise but a simple functional dependency from a hidden variable. In fact, it is hidden for a standard CF method since it cannot directly access this feature by construction. Hence, the conclusion is that this prediction method (*MF* but also the others) cannot exploit the additional information brought by this feature and cannot effectively deal with the influence of this variable. Conversely, using item splitting, which is going to check the dependency of the ratings from contextual variables, we can improve the performance of *MF* (if $\alpha > 0.3$) using all the three mentioned splitting criteria. Note that the three splitting criteria have different behaviors when α increases.

t_{mean} and t_{prop} decrease the error also when α is small, however, when $\alpha > 0.5$ t_{IG} outperforms the other two splitting methods. The performance increases substantially when more and more items' ratings are influenced by this contextual feature c .

Finally, Figure 3.5(b) shows MAE computed only for test ratings belonging to the items that were actually split. The figure shows that for these items item splitting is more and more effective with increasing values of α , i.e., when more ratings are influenced by the feature c . We observed that MF benefits from the item splitting for all the values of α . When α is small the differences are small, however, when the feature c gets more influential, then the pre-processing of the items pays off. The behavior of KNN is different (as before for the original Yahoo! data). This method benefits from item splitting only when α is getting larger, i.e., $\alpha > 0.4$.

3.2.4 Changes in Recommendation List

The majority of the previous works on context-aware recommender systems used contextual information to improve the accuracy of the rating prediction [Adomavicius et al., 2005, Anand and Mobasher, 2006]. However, to our best knowledge nobody analyzed how much the context actually impacts on the recommendation list, and if the changes in the items actually recommended to the user is really noticeable. We made an off-line experiment where we simulated the change in the context and computed the change in the recommendation list. Imagine, for instance, that the user is presented with a recommendation list in a day with good weather ($c = 0$). But suddenly the weather changes from good to bad ($c = 1$). We would like to know if, and to what extent, the recommendation list would change and if the user would notice the difference. To measure differences between top-k ranked lists we choose to measure average overlap between the lists. We believe that it gives a simple indication of the changes caused by the context but we agree that a measure such as NDPM [Yao, 1995] could have been also used and it is even more precise in comparing to ranked lists. We opted for the average overlap as it is very intuitive measure and our primary goal was to get an intuition about how much context could influence the top-k ranked list. We haven't see any research that would compare the differences of the lists themselves.

To evaluate the amount of change in the recommended item list we used three semi-synthetic data sets with increasing number of modified ratings: $\alpha = 0.1\beta = 0.1$, $\alpha = 0.5\beta = 0.5$, $\alpha = 0.9\beta = 0.9$. These three data sets play the role of the historical data used as a training set and therefore are fixed. As in the previous experiments with semi-synthetic data sets we used context C consisting of two features: user age information and the artificial contextual feature c . For the prediction we used item splitting with T_{IG} splitting criteria and set the threshold

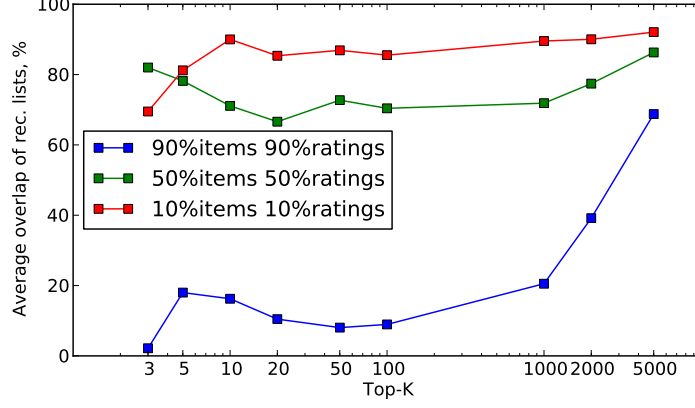


Figure 3.6: Average overlap of recommendation list when context changes.

to 0.01. When making the prediction for each of the $user \times item$ pair, we fixed the target context $c = 1$ and built a recommendation list of the items in a standard way described in Section 3.1. To simulate the change in the context, we set the target context $c = 0$ and built a second recommendation list. Then we compared these two recommendation lists.

The obtained results for a sample of 200 randomly chosen users are shown in Figure 3.6. Here we measure the average overlap of two recommendation lists normalized by the list length. For example, when $\alpha = 0.5, \beta = 0.5$ then the top-5 recommendation lists produced in the two contextual conditions overlap by 80%, that corresponds to the change of 1 item in the recommendation list. Note, that a reasonable recommendation list does not contain normally more than 10 items. Therefore, changes in longer recommendation lists are mentioned here for sake of completeness. The results show that even in the data sets where context has a minor impact, the recommendation list is really changing as effect of the changed contextual condition. As it was expected, the higher is the impact of the contextual variable, the bigger is the change. In top-3 recommendation lists and $\alpha = 0.9, \beta = 0.9$, all the recommended items will change as an effect of the change of the contextual feature c .

This comparison does not take into account how accurate the predictions are. However, our previous experiments confirmed that on average when taking into account contextual information, the prediction accuracy improves. Hence we believe that user would notice the changes in the recommendations and he would benefit from a context-aware recommender system. The final assessment should be done after a user study and it is part of our future work.

Missing values	0%	20%	40%	60%	80%	100%
MAE	0.706777	0.804095	0.880746	0.925491	0.932382	0.930368

Table 3.2: Item splitting performance with an increasing number of missing contextual values.

3.2.5 Performance with Missing Contextual Values

This section evaluates the performance of the proposed algorithm as the number of missing contextual values increases. We performed the experiment using $\alpha = 0.9, \beta = 0.9$ semi-synthetic data set. An increasing fraction of the artificial contextual feature values were replaced by the unknown value. We used the method described in Subsection 3.1.3. Table 3.2 summarizes the results.

The accuracy of the algorithm drops when we reduce the amount of known information about the contextual conditions. We want to notice that the accuracy is not a linear function of the amount of missing information. When data set contains only 40% of known values, the accuracy is approximately as low as the context free prediction. When all values are missing, algorithm performs the same as the baseline predictor that does not take into account context. This could be explained by looking more carefully at how the algorithm deals with missing values. In fact, when all the contextual values are missing in the training set the algorithm does not split any item. This situation arises because the algorithm assigns the ratings to both the newly introduced items. In such case both newly generated items are identical and all the statistical tests do not capture any significant difference between these items. In conclusion, for our used data set item splitting method benefits from the contextual information if we have at least half of the contextual feature values present.

3.2.6 Precision and Recall Analysis

To understand the potential of item splitting in a context-dependent set of ratings we tested this approach on the semi-synthetic data sets described earlier, i.e., replacing the gender feature with a new contextual variable that does influence the ratings. The baseline method is again *MF* when no contextual information is considered. This is compared again with Reduction Based [Adomavicius et al., 2005], and our item splitting technique. Figure 3.7 shows the comparison of these three methods on nine semi-synthetic data sets. For each data set we computed precision and recall. We considered item as worth recommending if the algorithm made a prediction greater or equal to 4. For all the nine data sets the algorithm splits an item if the split leads to an IG bigger than 0.01.

As we expected, the smaller is the impact of the contextual feature c , the

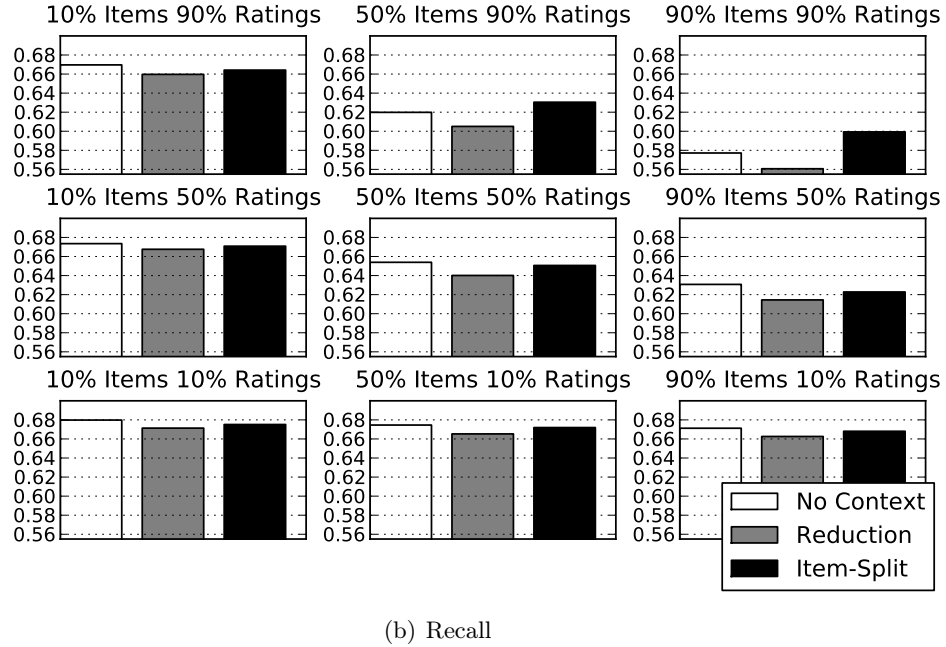
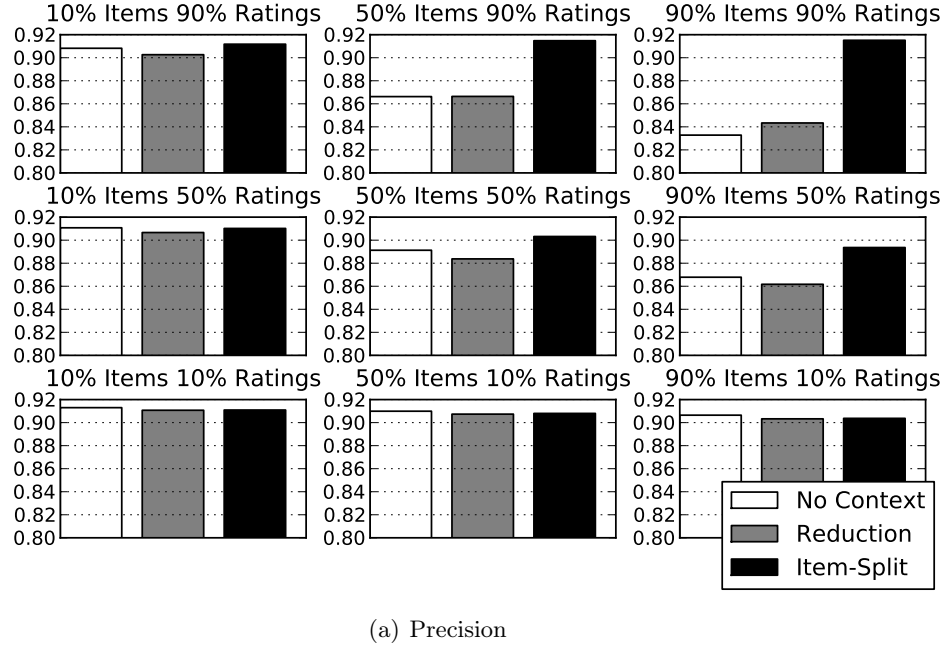


Figure 3.7: Comparison of contextual pre-filtering methods.

smaller is the improvement of the performance measure obtained by the methods that exploit the context. In fact, item splitting improved the performance of the baseline method in 4 cases: $\alpha \in \{0.5, 0.9\}, \beta \in \{0.5, 0.9\}$. The highest improvement in precision, 9.9%, was observed when $\alpha = 0.9, \beta = 0.9$, i.e., when most of the items and most of the ratings were influenced by the artificial contextual feature.

Reduction based approach increased precision by 1.3% only when $\alpha = 0.9, \beta = 0.9$. This is the data set where the artificial contextual feature has the highest influence on the ratings and 90% of items are modified. In [Adomavicius et al., 2005] the authors optimized MAE when searching for the contextual segments where the context-dependent prediction improves the default one (no context). Here, we searched for the segments where precision and recall is improved and we used all the best performing segments to make the predictions. For example, Figure 3.7(a) shows the precision of reduction based. To conduct this experiment, the algorithm first sought (optimizes) the contextual segments where the precision is improved (using a particular split of train and test data). Then, when it has to make a rating prediction, used either only the data in one of these segments, i.e., if the prediction is for an item-user combination in one of the found segments, or all the data, i.e., if the item-rating is in one contextual conditions where no improvements can be found with respect to the baseline. Note, that in all the three data sets where $\alpha = 0.5, \beta \in \{0.1, 0.5, 0.9\}$ the results are similar to the baseline approach. In these cases the reduction based approach does consider the segments generated using the artificial feature. However, the data set was constructed in such a way that half of the items do not have ratings' dependencies on the artificial feature, and no benefit is observed.

Precision/Recall curve. In this paragraph we illustrate the precision/recall curves for the three selected methods. For these experiments we reused the three data sets mentioned above: $\alpha = 0.1, \beta \in \{0.1, 0.5, 0.9\}$. As was done previously, we set the IG threshold to 0.01. For the reduction based approach we optimized precision. The results can be seen in Figure 3.8. The left figure shows the results for $\alpha = 0.9, \beta = 0.5$ and the right figure for $\alpha = 0.9, \beta = 0.9$. We skip the $\alpha = 0.9, \beta = 0.1$ data set, as for this data set all three methods perform similarly. Each curve was computed by varying the threshold at which a recommendation is done. For example, all the methods obtained the highest precision when recommending the items that were predicted as rating 5. Note that the ground truth is that a recommendation is relevant if the user rated the item 4 or 5. In order to compute precision/recall curves we computed these metrics retrieving the items with increasing values of their predicted ratings; we considered the thresholds in this set: $\{1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$. Note that the previous simpler experiment (see Figure 3.7) was done setting the

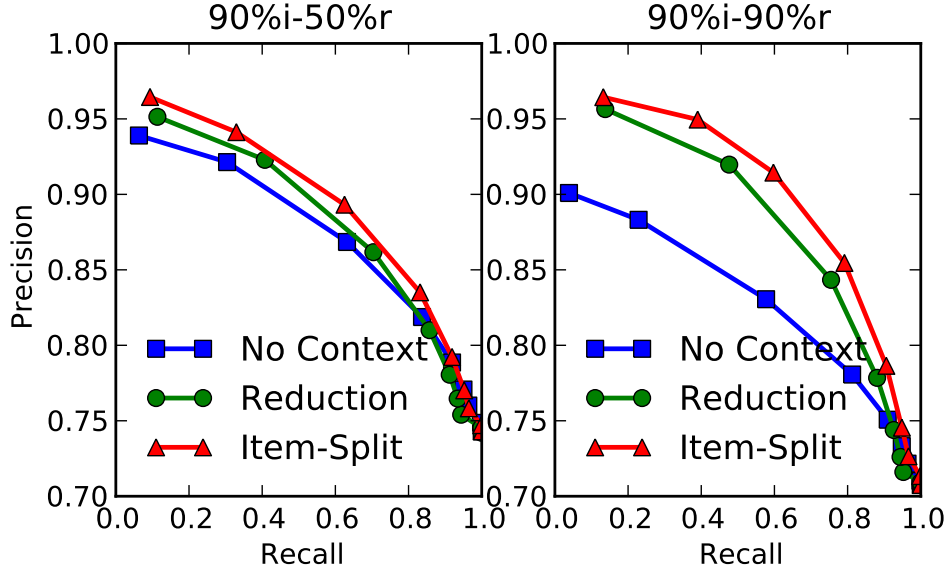


Figure 3.8: Precision/recall curves for two data sets.

recommendation threshold to 4, i.e., predicting as relevant the items whose predicted ratings are larger or equal to 4. The recall is clearly equal to 1 if we recommend all the items, i.e., those predicted with a rating of 1 and higher. Even at this level of recall, the precision is more than 70%. This can be explained by the high proportion of larger ratings in the data set.

Recommender systems usually try to improve precision more than recall, since the user is mostly interested in a small number of good recommendations, rather than in knowing what all the relevant items are. Having recall as small as 0.01, we could still be able to recommend too many items for user to consume, i.e., approximately 119 items in our data set. Interestingly, as we can see it is also much harder to make precise recommendations than to obtain high recall. The curves for all the three methods flat when approaching precision 0.97. At this point the system recommends only the items that were predicted with rating 5. This is the maximum possible predicted rating by *MF* and the precision can not be improved by varying the threshold at which recommendation is done. We also observe that we can achieve higher maximum precision for item splitting method comparing to other methods. When $\alpha = 0.9$ and $\beta = 0.9$, the highest precision for item split is 7% larger than the baseline method. The improvement when $\alpha = 0.9$ and $\beta = 0.5$ is 2.7%. This experiment gives valuable insights into the behavior of Reduction Based. We see, that for each recommendation threshold value Reduction Based shows a higher recall than the other two methods. At the highest precision level, Reduction Based is close to item splitting and improves

the precision of baseline method by 6.1% when $\alpha = 0.9, \beta = 0.9$ and by 1.3% when $\alpha = 0.9, \beta = 0.5$. But, the precision/recall curve of Reduction Based is always below the item splitting one.

In conclusion we want to note that considering both precision and recall, we see that both context-aware recommendation methods yields quite similar results. More noticeably, both methods outperforms baseline CF which does not take context into account.

3.3 Conclusions and Future Work

In this chapter we have provided a comprehensive evaluation of a contextual pre-filtering technique for CF, called item splitting. Based on the assumption that certain items may have different evaluations in different contexts, we proposed to use item splitting to cope with this situations. The method is compared with a classical context-aware pre-filtering approach [Adomavicius et al., 2005] which uses extensive searching in the space of the possible context-dependent segmentations of the data in order to find the contextual segments that improve the baseline prediction. We observed that despite the increased data sparsity, item splitting is beneficial, when some contextual feature separates the item ratings into two more homogeneous rating groups. However, if the contextual feature is not influential this splitting technique resulted in just a minor decrease of the prediction error on the split items and sometimes produced a minor increase of the error in the full data set.

We have also shown that item splitting outperforms reduction based context-aware approach when *MF* CF is used as rating prediction method. Moreover, item splitting is more time and space efficient and could be used with large context-enriched data bases. In this work we showed, that item splitting method would significantly change the recommendation list when the contextual conditions change. Smaller changes in the recommendation list are observable even when context has a small impact on the user evaluation.

Item splitting can be extended in several ways. For instance one can try to split the users (not the items) according to the contextual features in order to represent the preferences of a user in different contexts by using various parts of the user profile (micro profiling). Another interesting problem is to find a meaningful item splitting in continuous contextual domains such as time or temperature. Here, the splitting is not easily predefined but must be determined setting thresholds in the continuous space. Finally, item splitting could ease the task of explaining recommendations. The recommendation can be made for the same item in different contexts. The contextual condition on which the item was split could be mentioned as justifications of the recommendations. For example, we recommend you to go to the museum instead of going to the beach as it will

be raining today.

N-dimensional Tensor Factorization

Most model-based Collaborative Filtering approaches such as Matrix Factorization (*MF*) do not provide a straightforward way to integrate context information into the model. In this chapter, we introduce a CACF method based on *Tensor Factorization (TF)*, which generalizes Matrix Factorization, that supports a flexible and generic integration of contextual information. It models the data as a $User \times Item \times Context_1 \times \dots \times Context_k$ N -dimensional tensor instead of the traditional 2D user-item matrix. In the proposed model the different contextual features are considered as additional dimensions in the representation of the data as a tensor. The factorization of this tensor leads to a compact model of the data which can be used to provide context-aware recommendations. We propose here an algorithm to perform N -dimensional factorization, and we show that *TF* improves non-contextual Matrix Factorization up to 30% in terms of the Mean Absolute Error (MAE). We also compare *TF* to two state-of-the-art context-aware methods and show that Tensor Factorization consistently outperforms them in a semi-synthetic and a real-world data. The improvements range from 2.5% to more than 12% depending on the data. Noticeably, our approach outperforms other methods, e.g., by a wider margin whenever more relevant contextual information is available. To our best knowledge this is the first purely model based approach to context-aware collaborative filtering.

The main CF approaches that contributed most to the winning of the Netflix Prize ¹ were *neighborhood methods* and *latent factor models*. Neighborhood methods use similarity functions such as the *Pearson Correlation* or *Cosine Distance* to compute the sets of neighbors of a user or an item. Recommendations are then computed by using data from those neighbors. On the other hand, latent factor models [Basilico and Hofmann, 2004] such as *Matrix Factorization (MF)* solve the recommendation problem by decomposing the user-item matrix

¹www.netflixprize.com

and learning latent factors for each user and item in the data. The underlying assumption is that both users and items can be modeled by a reduced number of factors. This approach has been many times proven to be the most accurate CF method [Koren and Bell, 2010].

A tensor is the generalization of a two-dimensional matrix to a larger number of dimensions. In a tensor the usual user-item two-dimensional matrix is converted into a multi-dimensional tensor. Tensor Factorization can be used in a straightforward way to model any set of features that depend on the rating. In this work we focus on the use of TF for adding contextual information. We call our approach to contextual recommendation via TF as *Multiverse Recommendation* because it can be used to efficiently bridge hidden “worlds” separated by different contexts and therefore collapse parallel dimensions into a coherent model.

The relation of our model to other latent models and context-aware systems is reviewed in section 4.1, and our experimental results are summarized in section 4.3, where we validate the proposed approach on several synthetic and real world datasets. This proposed model brings a number of contributions to the area of contextual recommendations. First of all to our best knowledge this is the first pure model based approach to CACF. It includes the ability to:

- Generalize efficiently two-dimensional MF approaches to the N -dimensional case in a compact way;
- Include any number of contextual dimensions into the model itself;
- Train the model with a fast and straightforward algorithm.

4.1 Related Work

There has been a recent increase of interest in adding context to recommendations. This is probably due to the relevance of context in mobile applications and the success that some applications have had using contextual variables such as the user location (*e.g.* Foursquare ²).

The approach proposed in this chapter belongs to the contextual modeling algorithmic class of CARS (see Chapter 2 for details on CARS classification). Although some standard model-based approaches could theoretically accept more dimensions, the only models that have been experimented in this category is the Context-aware Support Vector Machine (SVM) proposed by Oku *et al.*'s in [Oku et al., 2006]. The authors consider support vectors in a multidimensional space and find the separating hyperplane. They later use this hyperplane to compute the user-to-user similarity that is used in the final prediction rule.

²<http://www.foursquare.com>

Their experiments show that contextual recommendations perform better than non-contextual.

Factorization models have recently become one of the preferred approaches to Collaborative Filtering, especially when combined with neighborhood methods [Koren, 2008]. However, even when processing standard datasets such as the one used for the Netflix Prize, the importance of context has been clearly shown. For instance, Koren has recently introduced a temporal model into the factor model called *timeSVD++* [Koren, 2009] that significantly improves the Root Mean Squared Error (RMSE) on the Netflix data compared to previous non-temporal factorization models.

We note that standard TF, including our proposed approach, assumes that each dimension of the tensor can take a finite number of discrete values. Therefore, to model continuous domains we first need to discretize them. For example, in context-aware CF the weather dimension could have 4 states: {sunny, rainy, cloudy, snowing}. Xiong *et al.* present a Bayesian Probabilistic TF which tries to overcome this limitation. The authors model the temporal evolution of online shopping preferences [Xiong et al., 2010]. Their experiments show that using time as the third dimension in the form of a tensor does improve accuracy when compared to the non-temporal case. As we will explain later, Xiong’s model can in fact be considered as a simple case instance of the more generic model presented in this chapter. Finally, three dimensional TF has also been proposed by Rendle *et al.*, for Collaborative tag Recommendation. Rendle *et al.* use the third tensor dimension in their *user, item, tag* model to represent the target variable which in their case are tags coded as binary vectors where the presence of a tag is marked by a 1 and the absence by a 0 [Rendle et al., 2009].

However to the best of our knowledge, there is no previous work on the use of N -dimensional tensors for CF. Moreover, to our best knowledge this is the first model based approach to CACF, which is the main contribution of this chapter and will be presented next in detail.

4.2 Multiverse Recommendation

TF is an existing N -dimensional extension of Matrix Factorization. In the current section we introduce the Matrix and Tensor Factorization models and explain the details of how we have adapted this model for the N -dimensional CF case.

The main idea behind the use of TF is that we can take advantage of the best properties of the Matrix Factorization to deal with N -dimensional information. However, before we dive into the details of the TF model, we shall briefly summarize the two-dimensional MF approach.

4.2.1 Matrix Factorization

MF assumes that ratings provided by users on items can be represented in a sparse $m \times n$ matrix $Y \in \mathbb{R}^{m \times n}$ where n is the number of items and m the number of users. The CF problem then boils down to matrix completion that in MF is obtained by factorizing the matrix of observed ratings into two matrices $U \in \mathbb{R}^{m \times d}$ and $M \in \mathbb{R}^{n \times d}$ such that $\mathbb{R}^{m \times n} \ni F := UM^\top$ approximates Y , i.e. minimizes a loss function $L(F, Y)$ between observed and predicted values. In most cases, a regularization term for better generalization performance is added to the loss function and thus the objective function becomes $L(F, Y) + \Omega(F)$. Standard choices for L include the least squares loss function $L(F, Y) = \frac{1}{2}(F - Y)^2$ and for Ω the Frobenius norm i.e. $\Omega(F) = \lambda [\|U\|_{\text{Frob}}^2 + \|M\|_{\text{Frob}}^2]$. Frobenius norm of matrix U is defined as the root square of sum of squares of the elements: $\|U\|_{\text{Frob}}^2 = \sqrt{\sum_i \sum_j |U_{ij}|^2}$.

4.2.2 Tensor Factorization

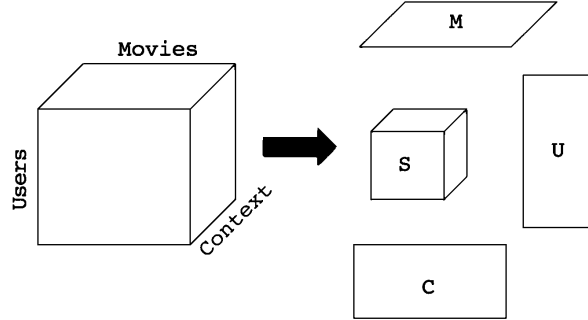


Figure 4.1: Illustration of the (3-dimensional) HOSVD tensor factorization model.

N -dimensional Tensor Factorization is a generic model framework for recommendations that is able to handle N -dimensional rating data, i.e. each rating can be associated with n features such as contextual dimensions, user and item. It shares most of the advantages of MF, such as fast prediction computations and simple and efficient optimization techniques. We shall now introduce the generic TF model and describe its specific adaptation to context-aware CF in Section 4.2.3.

Notation For the sake of simplicity, we will describe the model for a single contextual variable C , and therefore Y , the tensor containing the ratings, will be a 3-dimensional tensor. The generalization to larger numbers of contextual

dimensions is straightforward. In the following we denote the sparse tensor of rating observations by a $m \times n \times c$ tensor $Y \in \mathbb{R}^{m \times n \times c}$, where n denotes the number of items, m is the number of users, and c is cardinality of $\{v_1, \dots, v_c\}$, the set of values that context C can take. Typically, the rating is given on a 5-point Likert scale and thus $Y \in \{0, \dots, 5\}^{m \times n \times c}$, where the value 0 indicates that a user did not rate an item. In this sense, 0 is a special value since it does *not* indicate that a user dislikes an item but rather that data is missing. We will use two tensor operations:

- A tensor-matrix multiplication operator. The different dimensions of the tensor are called modes. A 1-mode tensor-matrix multiplication operator of tensor $S \in \mathbb{R}^{m \times n \times c}$ by a matrix $U \in \mathbb{R}^{m' \times m}$, denoted \times_1 is a $m' \times n \times c$ tensor in which the entries are given by

$$(S \times_1 U)_{j,i_2,i_3} = \sum_{k=1}^m s_{k,i_2,i_3} u_{j,k}$$

In our work instead of numbering the modes we use letters. For example, in \times_U notation, the subscript indicates that the user mode, which corresponds to the 1-st dimension of S , is the dimension used in the multiplication.

- The tensor outer product (also known as Kronecker product) denoted by \otimes . The tensor outer product of two matrices U and V is computed as

$$U \otimes V = \begin{bmatrix} u_{1,1}V & u_{1,2}V & \cdots \\ u_{2,1}V & u_{2,2}V & \\ \vdots & & \ddots \end{bmatrix}$$

Here $u_{i,j}V$ denotes scalar $u_{i,j}$ multiplication with a matrix V , resulting in a new matrix. Similarly, the matrix outer product with a vector would produce a tensor.

Finally, $D \in \{0, 1\}^{n \times m \times c}$ is a binary tensor with a nonzero entry D_{ijk} whenever Y_{ijk} is observed and we denote by U_{i*} the entries of the i_{th} row of matrix U .

HOSVD-decomposition There are different types of tensor decomposition models in the literature, such as the Canonical Decomposition or Parallel Factors – which is also known as the CP-decomposition – and the High Order Singular Value decomposition (HOSVD). In our approach, we follow the HOSVD formulation shown in Figure 4.1, where the 3-dimensional tensor is factorized into three matrices $U \in \mathbb{R}^{m \times d_U}$, $M \in \mathbb{R}^{n \times d_M}$ and $C \in \mathbb{R}^{c \times d_C}$ and one central tensor $S \in \mathbb{R}^{d_U \times d_M \times d_C}$. In this case, the decision function for a single *user* i , *item* j , *context* k combination becomes:

$$F_{ijk} = S \times_U U_{i*} \times_M M_{j*} \times_C C_{k*} \quad (4.1)$$

This decomposition model allows for full control of the dimensionality of the factors for the users, items and context by adjusting the d_U , d_M and d_C parameters. This property is valuable in the case of large real world datasets where the matrices U and M can grow in size and potentially pose a storage problem.

4.2.3 Tensor Factorization for CF

The aim of an N-dimensional TF approach to context-based recommendation is to model the context variables in the same way as the users and items are modeled in MF and to take into account the interactions between users-items-context. We shall refer to this approach as *Multiverse Recommendations*. Existing HOSVD methods (e.g. [Lathauwer et al., 2000]) require a dense matrix Y and therefore can not be applied to tensors generated by a recommendation problem that it is normally very sparse. Treating Y as a dense tensor with missing entries represented as 0, would introduce a bias in favor of unobserved ratings. In other words, the vast majority of the tensor entries would be zero and the corresponding HOSVD model would tend to predict that all the entries in Y as zeros. In fact, this solution would minimize the global loss function computed also on the missing values. The model of Regularized TF introduced in this section avoids this problem by optimizing the prediction only for the observed values in the rating tensor.

Multiverse Recommendations have a number of advantages compared to current context-based methods, including:

- No need for pre- or post-filtering. In contrast to many of the current algorithms which rely on splitting and pre- or post-filtering the data based on context, TF utilizes all the available ratings to model the users and the items. Splitting or pre-or post- filtering the data based on the context can lead to a loss of information about the interactions between the different context settings.
- Ability to handle N-dimensions. The TF approach we have introduced generalizes well to an arbitrary amount of context variables while adding relatively little computational overhead.

4.2.4 Loss Function

In analogy to MF approaches [Srebro et al., 2005, Weimer et al., 2008], we define the loss function as:

$$L(F, Y) := \frac{1}{\|D\|_1} \sum_{i,j,k} D_{ijk} l(F_{ijk}, Y_{ijk}) \quad (4.2)$$

where $l : \mathbb{R} \times \mathcal{Y} \rightarrow \mathbb{R}$ is a point wise loss function penalizing the distance between an estimate and an observation and F_{ijk} is given by equation 4.1. Note that the total loss L is defined only on the observed values in the tensor Y . Possible choices for the loss function l include the following:

Square error:

$$l(f, y) = \frac{1}{2}(f - y)^2 \text{ and} \\ \partial_f l(f, y) = f - y$$

Absolute loss:

$$l(f, y) = |f - y| \text{ and} \\ \partial_f l(f, y) = \text{sgn}[f - y]$$

Square error is a loss function that incorporates both the variance of the estimator and its bias, moreover, heavily weights outliers. Absolute loss estimates the average error that the model makes.

4.2.5 Regularization

Simply minimizing a loss function is known to lead to overfitting. Given the factors U, M, C, S which constitute our model, there are several approaches to ensure that the model complexity does not grow without bound. A simple option is to add a regularization term based on the l_2 norm of these factors [Srebro and Shraibman, 2005]. In the case of a matrix, this norm is also known as the Frobenius norm:

$$\Omega_{umc}[U, M, C] := \frac{1}{2} \left[\lambda_U \|U\|_{\text{Frob}}^2 + \lambda_M \|M\|_{\text{Frob}}^2 + \lambda_C \|C\|_{\text{Frob}}^2 \right]. \quad (4.3)$$

Here, λ_X is the parameter that control the relative importance of the term $\|X\|_{\text{Frob}}^2$. In a similar manner, we can also restrict the complexity of the central tensor S by imposing a l_2 norm penalty:

$$\Omega_s[S] := \frac{1}{2} \left[\lambda_S \|S\|_{\text{Frob}}^2 \right]. \quad (4.4)$$

In this work we use both regularization terms Ω_{umc} and Ω_s described in eq. 4.3 for U, M, C and eq. 4.4 for S .

4.2.6 Optimization

Overall, we strive to minimize a regularized risk functional that is a combination of $L(F, Y)$, $\Omega_{umc}[U, M, C]$ and $\Omega_s[S]$. The objective function for the minimization problem is:

$$R[U, M, C, S] := L(F, Y) + \Omega_{umc}[U, M, C] + \Omega_s[S] \quad (4.5)$$

Minimizing this objective function can be done using many approaches. Alternating subspace descent [Rennie and Srebro, 2005] is a popular choice in MF and could be used also here. In subspace descent one optimizes iteratively individual components of the model while keeping the remaining components fixed, *e.g.* optimize over the U matrix while keeping the remaining matrices and tensor fixed, then over M etc. This method quickly converges but requires the optimization procedure to be run in a batch setting, i.e. optimizing individual component using the full training set.

As the dataset size grows, it becomes increasingly infeasible to solve factorization problems by batch optimization. Instead, we resort to a simple online algorithm which performs stochastic gradient descent (SGD) in the factors U_{i*} , M_{j*} , C_{k*} and S for a given rating Y_{ijk} simultaneously. In order to compute the updates for the SGD algorithm, we need to compute the gradients of the loss function and eventually the objective function with respect to the individual components of the model:

$$\begin{aligned} \partial_{U_{i*}} l(F_{ijk}, Y_{ijk}) &\equiv \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) S \times_M M_{j*} \times_C C_{k*} \\ \partial_{M_{j*}} l(F_{ijk}, Y_{ijk}) &\equiv \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) S \times_U U_{i*} \times_C C_{k*} \\ \partial_{C_{k*}} l(F_{ijk}, Y_{ijk}) &\equiv \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) S \times_U U_{i*} \times_M M_{j*} \\ \partial_S l(F_{ijk}, Y_{ijk}) &\equiv \partial_{F_{ijk}} l(F_{ijk}, Y_{ijk}) U_{i*} \otimes M_{j*} \otimes C_{k*} \end{aligned}$$

The *Multiverse Recommendation* TF method is summarized in Algorithm 1, which is easy to implement since it accesses only one row of U , M , and C at a time. In addition, it is easy to parallelize by performing several updates independently. Note that the algorithm scales linearly to the number of ratings K . The overall algorithm training complexity is $O(Kd_U d_M d_C)$. The computationally expensive part of the algorithm is to make a single rating prediction which involves computing the outer tensor product (see Equation 4.1). The complexity of computing a single rating prediction for 3-dimensional case is $O(d_U d_M d_C)$. The complexity grows exponentially while adding new contextual dimensions. For example, having 2 contextual dimensions C_1 and C_2 the complexity of the rating prediction step becomes $O(d_U d_M d_{C_1} d_{C_2})$.

```

Input  $Y, d$ 
Initialize  $U \in \mathbb{R}^{m \times d_U}, M \in \mathbb{R}^{n \times d_M}, C \in \mathbb{R}^{c \times d_C}$  and  $S \in \mathbb{R}^{d_U \times d_M \times d_C}$  with
small random values.
Set  $t = 1$ 
for  $epoch = 1$  to  $10$  do
  while  $(i, j, k)$  in observations  $Y$  do
     $\eta \leftarrow \frac{1}{\sqrt{t}}$  and  $t \leftarrow t + 1$ 
     $F_{ijk} \leftarrow S \times_U U_{i*} \times_M M_{j*} \times_C C_{k*}$ 
     $U_{i*} \leftarrow U_{i*} - \eta \lambda_U U_{i*} - \eta \partial_{U_{i*}} l(F_{ijk}, Y_{ijk})$ 
     $M_{j*} \leftarrow M_{j*} - \eta \lambda_M M_{j*} - \eta \partial_{M_{j*}} l(F_{ijk}, Y_{ijk})$ 
     $C_{k*} \leftarrow C_{k*} - \eta \lambda_C C_{k*} - \eta \partial_{C_{k*}} l(F_{ijk}, Y_{ijk})$ 
     $S \leftarrow S - \eta \lambda_S S - \eta \partial_S l(F_{ijk}, Y_{ijk})$ 
  end while
end for
Output  $U, M, C, S$ 

```

Algorithm 1: Tensor Factorization

This could be a major problem if many contextual dimensions are considered. However, usually context-enriched data sets have a small number of dimensions (up to 10) and d_{C_j} is small, making the algorithm practical. However, in order to overcome this problem we are considering other tensor factorization methods that scale linearly to the number of dimensions. For instance, the Candecomp – Parafac (also known as Canonical Decomposition) [Hitchcock, 1927] method decomposes a tensor into matrices, without the central tensor S . Therefore, the prediction step can be done in a linear time.

4.2.7 Missing Context Information

TF also supports a straightforward approach to dealing with missing context information. Assume that we are missing the context information of a rating done by user i' on item j' $Y_{i',j'}$. Intuitively, one would like to add the rating information in the profile information of the user and the item without updating the information of the context profile or applying the update equally on all context profiles. There are thus two options:

- Update the $U_{i'*}$ and $M_{j'*}$ factors and skip the C and S factors update. Intuitively, it would use a data tuple to update the user and the item model, but will not learn anything about the context;
- Update the $U_{i'*}$ and $M_{j'*}$ factors while updating all the context profiles in C , but with a step size η divided by the number of context cases c . We can then update the central tensor S using all the context cases dividing

the step size by c . Intuitively, it would update user and item model with a new data tuple. Moreover, it will update context model while assuming that this tuple was done with a uniform distribution in all the possible contextual conditions.

4.3 Experiments

This section provides the experimental evaluation of our proposed *Multiverse Recommendation* TF algorithm. First, we analyze the impact of using contextual information by comparing the algorithm to standard non-context-aware MF. Then we compare TF to two state-of-the-art context-based collaborative filtering algorithms presented in [Adomavicius et al., 2005, Baltrunas and Ricci, 2009a]. We evaluate the algorithms on two real world datasets, one of them kindly provided by Adomavicius *et al.* [Adomavicius et al., 2005]. Moreover, in order to better understand the behavior of the methods, we use a synthetic dataset where we control the influence of the context variable.

Before reporting the results, we will now illustrate the details of the experimental setup including the datasets used, the experimental protocol, and the different context-aware methods we compare against.

4.3.1 The data

We have tested the proposed method on two real world and 6 semi-synthetic data sets with ratings in a $\{1, \dots, 5\}$ scale. The semi synthetic datasets were generated using the Yahoo! Webscope movies data set. The second dataset is derived from the one used by Adomavicius *et al.* [Adomavicius et al., 2005]. It contains 1464 ratings by 84 users for 192 movies. In our experiments we refer to this data set as *Adom*. The third dataset was used by Hideki Asoh *et al.* [Ono et al., 2009] and provided by the authors. It contains food rating data from 212 users on 20 food menus. More information on these data sets can be found in Appendix C on page 155.

4.3.2 Evaluation Protocol

We have assessed the performance of the models by conducting a 5-fold cross-validation and computing the *Mean Absolute Error (MAE)*, which is one of the most widely used performance measures in the recommender systems literature [Herlocker et al., 2004].

We used the *Absolute loss* function as defined in 4.2.4. It is important to note that the TF approach allows for direct optimization of the evaluation measure, i.e., when using *MAE* the ideal loss function for minimizing this measure is the Absolute loss function. However, if the error measure is *RMSE*, we should

optimize the *least-square loss* function. For the training of the TF algorithm we use 10 epochs.

We have regularized using the l_2 norm both the matrices and the central tensor as explained in Section 4.2.5. Due to computational and time constraints we only conducted a limited search for parameters' optimization on all methods tested. For the TF algorithm, we set the regularization parameters $\lambda = \lambda_U = \lambda_M = \lambda_C$ and we thus end up with two regularization parameters λ and λ_S , the initial learning rate, and the dimensions of the individual components of the models d_U , d_M etc. as parameters.

All the reported results are the average of a 5-fold cross-validation. We do not report the variance of the results since it was insignificant in our experiments, and did not qualitatively influence our findings and conclusions. Moreover, all differences between TF and the other methods are statistically significant.

4.3.3 Alternative Context-based Methods

We choose two state-of-the-art context aware collaborative filtering methods that are based on pre-filtering and compare them to N -dimensional Multiverse Recommendation based on TF.

The first one is a *reduction* based approach, which is based on OLAP [Adomavicius et al., 2005], and extends classical Collaborative Filtering methods by adding contextual information to the representation of users and items. This reduction based method computes recommendations using *only* the ratings made in the same context as the one of the target recommendation. The exact contextual segments that optimize the prediction are searched (optimized) among those that improve the accuracy of the prediction. This is a rather computationally expensive operation, as for each combination of the contextual conditions, a Collaborative Filtering model needs to be trained and tested.

The second method that we use in our experiments is *item splitting*, described in chapter 3. In short, item splitting identifies items that have significant differences in the ratings [Baltrunas and Ricci, 2009a]. For each of these items, it splits the ratings into two subsets, creating two new artificial items with ratings assigned to these two subsets. The split is determined by the value of one contextual variable c_j , i.e., all the ratings that have been acquired in a context where the contextual feature c_j took a certain value.

Note, that both pre-filtering methods use a standard MF approach as the main CF algorithm. Moreover, both methods exploit the tradeoff between *less* and *more relevant* data and thus increase data sparsity. In reduction method, sparsity increases as it filters out irrelevant ratings. In item-splitting we introduce more specialized items for the same number of ratings. This is not the case for the TF model where all the data is used to model users, item and context factors.

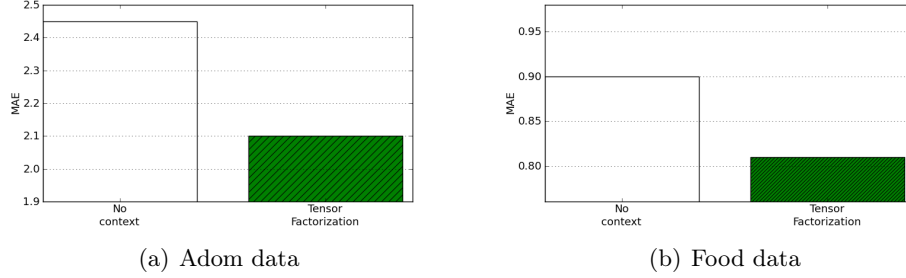


Figure 4.2: Comparison of matrix (no context) and tensor (context) factorization on the Adom and Food data.

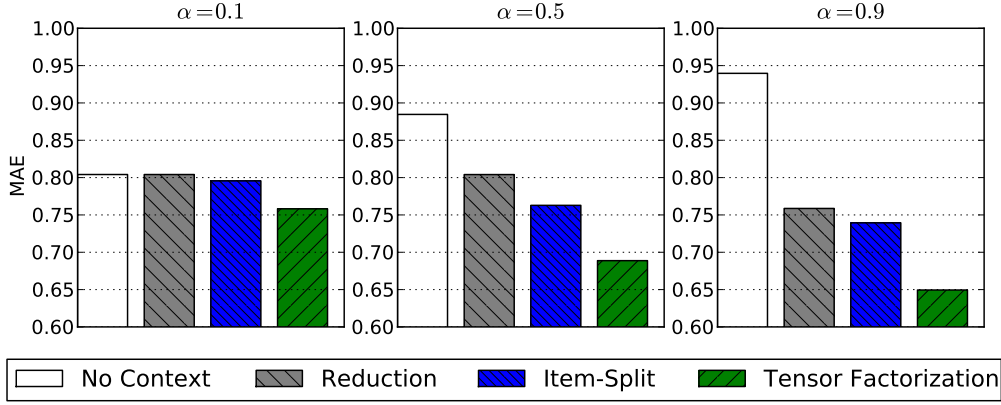


Figure 4.3: Comparison of context-aware methods on the Yahoo! artificial data ($\beta = 0.9$)

4.4 Results

We have first conducted some experiments to assess the relevance of contextual information. In order to do so, we compared our TF method with a regular non-context-aware MF method. Then we measured the goodness of our approach by comparing it to the other context-aware methods explained in Section 4.3.3 on different datasets and contextual conditions.

4.4.1 Tensor vs. Matrix Factorization

In this section we compare the N -dimensional TF approach to standard MF. That is, we use all the available contextual information in the TF model while we limit the MF approach to the standard user-item-rating setting, not adding

any contextual variables.

Figure 4.2 shows the results of Tensor and Matrix Factorization on the real world Adom and Food data. We observe that adding information in the form of context variables does make a significant (14% for the Adom and 10% for the Food data) difference in the performance. Note that when we use the Adom and Food data we end up with a 7-Dimensional and a 4-Dimensional TF models respectively since the Adom data contains 5 contextual variables and the Food data 2.

Figure 4.3 depicts the results of Tensor (in green) and Matrix Factorization (in white) on the artificial Yahoo dataset. As expected, the TF model outperforms the standard non-context aware MF method by 5% in the low context data ($\alpha = 0.1$) up to 30% in the high context case ($\alpha = 0.9$). When the context variable is not used in the Collaborative Filtering model, the contextual information acts as noise added to the data. A non-contextual model such as MF cannot distinguish between the noise and the functional dependency from a hidden variable. Since we are collapsing all the information into a standard two-dimensional MF model, the method fails to model this influence. In fact, MF alone cannot exploit the additional information contained in this feature and cannot effectively deal with the influence of this variable. We observe that the stronger the influence of the context variable the higher the MAE for MF.

4.4.2 Comparison to Context-Aware Methods

We now compare the pre-filtering based context-aware methods to TF. Figure 4.3 shows a comparison of the TF method with the various context-aware CF methods. The higher the influence of the context variable, the better the TF methods performs compared to the other context-aware methods. Also note that when “context” has a strong influence ($\alpha \geq 0.5, \beta \geq 0.9$), all the three context-aware methods outperform the MF method by 4% in the low influential context case, and up to 12% in the highly influential context data set ($\alpha = 0.9$). The performance advantage of the context-based method increases substantially when more information about the rating is encoded by the contextual feature c . The biggest improvement is observed when $\alpha = 0.9, \beta = 0.9$. In fact, for this data set the reduction-based approach improved MAE by 23.9%, item splitting improved by 24.2%, TF by 30.8% compared to the non-Context model.

More notably, for all the 3 synthetic datasets TF uniformly outperforms the other methods. We also observe that as the influence of context increases, so does the gain when using TF compared to both the reduction-based approach and item splitting. The proposed *Multiverse Recommendation* method also has the overall best performance when $\alpha = 0.9, \beta = 0.9$. It seems to efficiently exploit the linear dependency between the ratings and the contextual feature. We also observe that the performance of the non-context-aware MF model deteriorates

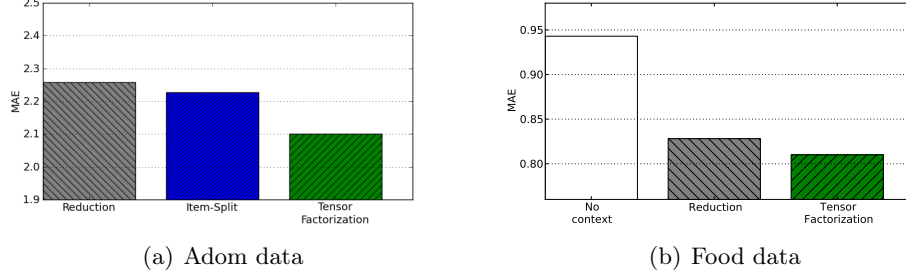


Figure 4.4: Comparison of context-aware methods on the Adom and Food data.

with the growing influence of the context variable. As mentioned above, this is due to the fact that without the context variable included in the model, the influence of the context in the data acts as a source of noise that cannot be modeled by simple MF.

We carried out a second set of experiments using the described real world data – Adom and Food (see Section 4.3.1 for more details). Figure 7.6 compares the same methods used in the previous experiment. The best performing method for these datasets is again TF, that outperforms both contextual pre-filtering methods and the context free method (MF) by at least 4.5%. For the Adom data we observe that TF outperforms both context-aware methods while for the food data we compared against the reduction based method which again is outperformed by the TF method by 2%.

4.5 Conclusions

Matrix Factorization is one of the most popular approaches to Collaborative Filtering but the model is not flexible enough to exploit the information coming from contextual dimensions in a straightforward manner. We have presented here an extension of this model using tensors. We have adapted the generic Tensor Factorization approach to Collaborative Filtering by adding a regularization component and we have shown how this can be used to embed multiple contextual dimensions into a coherent Multiverse Recommendation model.

In the experimental results with three datasets, we have obtained higher accuracy by taking into account the contextual variables. When comparing TF to standard non-contextual MF, we measured gains that range from 5% to almost 30% both in semi-synthetic and real-world data. We have also shown that TF consistently outperforms current state-of-the-art context-aware recommendation approaches, with performance gains ranging from 2.5% to more than 12%. Furthermore, we have seen that the relative gain is proportional to the amount of

relevant contextual information available.

We believe that Multiverse Recommendations open up a new avenue for recommender systems and we plan to further investigate extensions of TF models for recommender systems. In particular, we are interested in investigating the use of this model to further explore temporal dependencies in standard CF settings while also dealing with implicit feedback. We also plan to explore how multidimensional TF can be used to model non-contextual variables such as those related to content and user.

Group Recommendations

The large majority of RSs are designed to make recommendations for individual users. Since recommendations are usually personalized, different users receive diverse suggestions. These suggestions can be further contextualized, i.e. adapted to the current context of the user and the item. However, in some circumstances the items to be selected are not intended for personal usage but for a group of users; e.g., a DVD could be watched by a group of friends or in a family. Here, the group plays a special role, as a kind of companion context, and therefore the recommendations should be adapted not only to the user preferences, but also to the preferences of each group member. These groups can vary from stable groups to ad hoc groups requiring recommendations only occasionally. Group recommendations usually involve a trade-off between individual preferences. It is commonly understood that the group recommendations could not be, for each group member, as effective as those generated specifically for a user.

The composition of the group of users that are supposed to use a recommended item is a major contextual factor in many domains such as tourism or movies. In Chapter 3 and Chapter 4 we discussed general CACF algorithms. These CACF methods can not be directly used to solve a group recommendation problem. In group recommendation the number of possible contextual values is very large, i.e. each particular group is a unique context value. Therefore, in order to use the general CACF techniques one should collect ratings for each possible group requiring recommendations. This task is clearly impossible as the groups change dynamically and the number of items that a group experiences, and can rate, is usually small. Therefore, most of the research on group recommendations is concentrated on very specialized solutions. These analyze and exploits preferences of the group members.

This chapter analyzes the effectiveness of group recommendations obtained by aggregating the individual lists of recommendations produced by a CF system.

We compare the effectiveness of individual and group recommendation lists using normalized discounted cumulative gain. It is observed that the effectiveness of a group recommendation does not necessarily decrease when the group size grows. Moreover, when individual recommendations are not effective a user could obtain better suggestions looking at the group recommendations. Finally, it is shown that the more alike the users in the group are, the more effective the group recommendations are. The details can be found in Section 5.3.

Some recent works have addressed the problem of identifying recommendations “good for” a group of users, i.e., trying to satisfy, as much as possible, the individual preferences of all the group’s members [Jameson and Smyth, 2007]. Group recommendation approaches are either based on the generation of an integrated group profile or on the integration of recommendations built for each member separately. Group RSs have been designed to work in different domains: web/news pages [Pizzutilo et al., 2005], tourism [McCarthy et al., 2006, McCarthy, 2002], music [Crossen et al., 2002, McCarthy and Anagnost, 1998], TV programs and movies [O’Connor et al., 2001, Yu et al., 2006a].

A major issue in this research area relates to the difficulty of evaluating the effectiveness of group recommendations, i.e., comparing the generated recommendations for a group with the true preferences of the individual members. One general approach for such an evaluation consists of interviewing real users. In this approach there are two options: either to acquire the users’ individual evaluations for the group recommendations and then integrate (e.g., averaging) these evaluations into a score that the group “jointly” assigns to the recommendations; or to acquire directly a joint evaluation of the group for the recommendations. In the first case one must decide how the individual evaluations are integrated; this is problematic as different methods will produce different results and there is no single best way to perform such an integration. Another difficulty, which is common to both options, as was observed by [Masthoff and Gatt, 2006], is related to the fact that the satisfaction of an individual is likely to depend on that of other individuals in the group (emotional contagion). Moreover, on-line evaluations can be performed on a very limited set of test cases and cannot be used to extensively test alternative algorithms.

A second approach consists of performing off-line evaluations, where groups are sampled from the users of a traditional (i.e., single-user) RS. Group recommendations are offered to group members and are evaluated independently by them, as in the classical single user case, by comparing the predicted ratings (rankings) with the ratings (rankings) observed in the test set of the user. The group recommendations are generated to suit simultaneously the preferences of all the users in the group and our intuition suggests that they cannot be as good as the individually tailored recommendations. We observe that using this evaluation approach, in order to test the effectiveness of a group recommendation, we do not need the joint group evaluations for the recommended items, and we

can reuse the most popular data sets (e.g., Movielens or Netflix) that contain just evaluations (ratings) of individual users.

In this chapter we follow the second approach evaluating a set of novel group recommender techniques based on rank aggregation. We first generate synthetic groups (with various criteria), then we build recommendation lists for these groups, and finally we evaluate these group recommendations on the test sets of the users. The group recommendations are built in two steps: first a collaborative filtering algorithm produces individual users' rating predictions for the test items and consequently individual ranking predictions based on these rating predictions are generated. Then a rank aggregation method generates a joint ranking of the items recommended to the group by integrating the individual ranking predictions computed in the previous step. We then measure, for each group member, how good this integrated ranking is, and if this is worse than the initial individual ranking built by the system by taking into account only the ratings contained in the profile of the user. We performed an analysis of the generated group recommendations (ranking) varying the size of the groups, the inner group members similarity, and the rank aggregation mechanism.

As we mentioned above, intuition suggests that aggregating individual rankings, i.e., mediating the potentially contrasting preferences of the group members, should result in a decrease of the recommendation effectiveness. Moreover, one can also conjecture that the larger is the number of people in the group the harder it is to find a consensus among them. In this work, we show that these intuitions are not always correct. In fact, we show that in certain cases group recommendations with rank aggregation can be more effective than individual recommendations, when the effectiveness is measured using normalized discounted cumulative gain (nDCG), a popular IR metric measuring the quality of the ranking produced by a system. We observe that this happens when the individual recommendations are not particularly good, i.e., when the recommender system is not able to make good personalized recommendations. We show that this happens often in real scenarios. Moreover, we confirm the intuition that the effectiveness of the group recommendations raises when the group members are more similar.

It is worth noting that in this chapter we focus on the goodness of the predicted *ranking* of the recommendations rather than on evaluating the accuracy of the predicted ratings' values. We believe that especially for group recommendations it is more important to understand if the RS correctly sorts the proposed items, hence suggests to the group first the items more suitable for all the group members. This approach follows a new trend of evaluating recommender systems [Weimer et al., 2007]. Moreover, we observe that the rank aggregation problem has been largely studied and the notion of optimal rank, i.e., the Kemeny optimal one, has been defined [Dwork et al., 2001]. Some of the rank aggregation techniques that we have used (e.g., Spearman footrule optimal aggregation)

are proved to be close to the Kemeny optimal one and therefore our results show important and fundamental properties of the best solutions to group recommendation via rank aggregation.

The rest of the chapter is structured as follows. Section 5.1 defines our approach based on rank aggregation. Section 5.2 explains off-line experimental setup. Section 5.3 provides the experimental evaluation of the approach and finally Section 5.4 draws the conclusions and describes some future work.

5.1 Rank Aggregation

Our group recommendation method is based on the ordinal ranking of items, i.e., the result of a group recommendation process is an ordered list of items. To generate the ranking we use rank aggregation methods, taking a set of predicted ranked lists, one for each group member, and producing one combined and ordered recommendations' list. Most of the available rank aggregation methods are inspired by results obtained in Social Choice Theory [Arrow, 1950]. This theory studies how individual preferences could be aggregated to reach a collective consensus. Social choice theorists are concerned with combining ordinal rankings rather than ratings. Thus, they search for a societal preference result, where output is obtained combining several people ordered preferences on the same set of choices. For instance, in elections a number of ranked candidates, provided by voters, is collected and aggregated into one final ranked electors' list. Arrow [Arrow, 1950] proved that this aggregation task is impossible, if the combined preferences must satisfy a few compelling properties. His theorem states that there cannot be one perfect aggregation method, and this allows for a variety of different aggregation methods to exist.

In this chapter we will discuss a few different rank aggregation methods. There are two known ways to build a ranked list of recommendations for a group by using aggregation methods. The first approach can be applied to lists of items where a prediction of the user rating for each item is available. This approach computes first the group score for an item, which is a kind of prediction of the joint group rating, by integrating the predicted ratings for the item in the various input lists. Then the group score for the item is used to rank the items for the whole group. The other approach uses only the ranked lists produced for each individual and then in order to get the final group ranking applies aggregation methods on these lists.

Creating recommendations using rank aggregation methods addresses the problem of finding “consensus” ranking between alternatives, given the individual ranking preferences of several judges [Dwork et al., 2001]. The appropriate rank aggregation method usually depends on the ranking distance to optimize [Dwork et al., 2001] and also on the properties of the lists that are aggregated.

The aggregated list that minimizes the average Kendall tau distance from the input lists is called the Kemeny optimal aggregation. The Kendall tau distance counts the number of pair-wise disagreements between two lists. Kemeny optimal aggregation is the unique method that simultaneously satisfy natural and important properties of rank aggregation functions, called neutrality and consistency in the social choice literature (also known as Condorcet property) [Young and Levenglick, 1978]. However, it was proved in [Dwork et al., 2001] that such aggregation is NP-Hard and, therefore, we consider rank aggregation methods that approximate Kemeny optimal aggregation. These methods work with full lists of items, i.e., the aggregated lists contain the same items. We note that in our case this is not a limitation as we use collaborative filtering based on latent factor model [Koren, 2008]. Factor models can return ratings' predictions for all the unrated items and therefore can generate a full list of ranked items for all the users.

We now describe the rank aggregation methods that we use. All these methods take as input a set of items' permutations $g = \{\sigma_1, \dots, \sigma_{|g|}\}$, one for each group member, and produce a new permutation σ_g , i.e., the recommended ranking for the group g . σ_u is the permutation of the items $I = \{i_1, \dots, i_n\}$, representing the ranked list recommendations for user u ; $\sigma_u(i)$ is the position of item i in this list. For instance, $\sigma_u(i) = 1$ means that the item i is in the top position in the ranked list of recommendations for user u .

Spearman footrule aggregation finds an aggregation that minimizes the average Spearman footrule distance to the input rankings. The Spearman footrule distance between two lists is the sum, over all the items in I of the absolute difference between the rank positions of i in the two lists. For instance, if σ_u σ_v are two permutations of the items in I their Spearman footrule distance is: $F(\sigma_u, \sigma_v) = \sum_{i \in I} |\sigma_u(i) - \sigma_v(i)|$.

This aggregation method produces a ranking having average Kendall tau distance from the integrated rankings less than twice the average Kendall tau distance of the Kemeny optimal aggregation from the same rankings [Dwork et al., 2001]. This aggregation is computed by finding a minimum cost perfect matching in a particular bipartite graph. Computations are made on the weighted complete bipartite graph (I, P, W) as follows. The first set of nodes I denotes the set of items to be ranked. The second set of nodes P denotes the number of available positions. The weight $W(i, p)$ is the total footrule distance (from all the individual user rankings) of the ranking that places element i at position p , and is given by $W(i, p) = \sum_{u \in g} |\sigma_u(i) - p|$. We refer to [Dwork et al., 2001] for the details.

In **Borda count** aggregation method each individual item in a group member's ranked list is awarded with a score which is given according to its position in the list, $score_u(i) = n - \sigma_u(i) + 1$. The lower is the item position in the list (i.e., the larger is the value of $\sigma_u(i)$) the smaller is the score. Finally, the score points for the users in the group g , are added up, $score_g(i) = \sum_{u \in g} score_u(i)$,

and are used to produce the aggregated ranking (in decreasing group score value). Recently, it was shown by Coppersmith et al. [Coppersmith et al., 2006] that the Borda’s method produces aggregated lists that have average Kendall tau distance from the integrated rankings in g less than five times the average Kendall tau distance of the Kemeny optimal aggregation from the rankings in g .

The next two aggregation methods require that the items in the lists to aggregate are sorted according to (predicted) *ratings*. Since these lists represent recommendations for users the ratings are those predicted by the recommender system. The output of these methods is again a ranked list of items but the ranking is computed by exploiting an aggregation method that uses the (predicted) ratings.

In **Average** aggregation the item i group score is equal to the average of the predicted ratings for the individuals, i.e., $score_g(i) = \frac{\sum_{u \in g} \hat{r}_{ui}}{|g|}$, where \hat{r}_{ui} is the predicted rating for user u , item i combination. Then the ranking is computed accordingly (decreasing values of group score).

Least Misery strategy also uses the predicted ratings of the items in the individual recommendation lists. In this method the group score for item i is equal to the smallest predicted rating for i in the group, i.e., $score_g(i) = \min_{u \in g} \{\hat{r}_{ui}\}$. Thus, each item is predicted to be liked by the group as the less satisfied member, and again in the integrated group ranking the items are ordered according to these scores.

As a baseline we have also considered the **Random** aggregation method; it returns a random permutation of the items.

5.2 Experimental Setup

In the experimental evaluation we measure the effectiveness of the proposed group recommendation techniques. For each member of a group, we compare the effectiveness of the ranked recommendations generated for his group with the effectiveness of those computed only for him. Effectiveness of a ranking is computed with nDCG, as it will be illustrated below. We deliberately avoided to compute the joint effectiveness of the group recommendations for the group because, as we discussed in the introduction, it requires to define an arbitrary aggregation formula integrating the effectiveness of the group recommendation for all the group members.

To conduct the experiments we first generated artificial groups of users (see Subsection 5.2.1 for the details). Then, for evaluating the goodness of the recommendations (both individual and group) we used the standard approach to divide the dataset into two parts: the training set and the testing set. Approximately 60% (randomly chosen) of the ratings from the user profile were assigned to the training set and the rest 40% to the testing set. We used collaborative filtering to

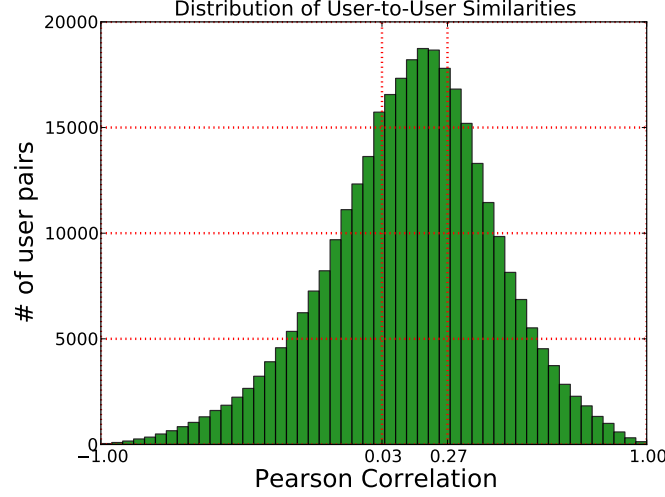


Figure 5.1: User-to-user similarity distribution.

generate individual predictions for each user (see Subsection 5.2.2 for the details). Then, these predictions were aggregated into the group recommendations using the previously described methods.

Finally, in the last step we computed the recommendation lists (individual and group) ranking precision using Normalized Discounted Cumulative Gain (nDCG) measure (see Subsection 5.2.3).

5.2.1 Data Set and Group Generation

To conduct the experiments, user groups of varying sizes and similarities were generated by sampling MovieLens data set, which contains 100K ratings for 1682 movies by 943 users (the precise number of generated groups depends on the experiments and it is reported later on). See Appendix C for details. We considered groups containing two, three, four and eight users. Moreover, we distinguished between *random groups* and groups with *high inner group similarity*. These are cases that are common in a real life situations. Groups with highly similar members represent people with common tastes, such as groups of friends. Whereas, random groups represent people without any social relations, such as random people taking the same bus.

Groups with high inner group similarity are defined as those containing users with user-to-user similarity higher than 0.27; where similarity is computed using Pearson correlation coefficient (PCC). In our data set 33% of all the possible user pairs have similarity higher than that threshold of 0.27 (see Figure 5.1).

Hence, for building a group with high inner group similarity we made sure that each user to user similarity in the group is higher than the threshold, i.e. fall into the top segment of data. Random groups were formed without considering any restriction on the user-to-user similarity. That led to a lower overall inner group similarity. For example, the average similarity of the users in these random groups is 0.132, whereas the average similarity of the users in the groups with high inner group similarity is 0.456.

When forming the groups, in order to measure if two users are similar or not we decided to consider only pairs of users that have rated at least 5 common items. This is a common practice in memory based CF literature and assures that the computed similarity value is reliable and the correlation is not high or low just by chance, i.e., because there is a perfect agreement or disagreement on a small set of items.

5.2.2 Recommendation Lists for a Group

To compute rating predictions for each user and generate individual recommendations' lists we used a popular model based collaborative filtering approach using matrix factorization with gradient descent optimization [Koren, 2008]. Hence, in our experimental setup individual predictions are computed using Singular Value Decomposition latent factor model with 60 factors. Using this prediction method for each user we generated a ranked list of recommendations containing all the items that are not present in the user's training set of any group member. Then, as we have discussed earlier, our group recommendation algorithm takes as input either these individual ranked lists of items' recommendations or the predicted ratings for each user in the group, and returns a ranked list of recommendations for the whole group. The individual recommendations are aggregated using the five methods described in Section 5.1.

5.2.3 Effectiveness of a Ranked List of Recommendations

For evaluating the goodness of a ranked list of recommendations we use a standard IR measure, i.e., Normalized Discounted Cumulative Gain (nDCG) [Manning, 2008]. Let p_1, \dots, p_l be a ranked list of items produced as an individual or a group recommendation. Let u be a user and r_{up_i} the true rating of the user u for the item p_i (ranked in position i , i.e., $\sigma_u(p_i) = i$). Discounted Cumulative Gain (DCG) and normalized DCG (nDCG) at rank k are defined respectively as:

$$DCG_k^u = r_{up_1} + \sum_{i=2}^k \frac{r_{up_i}}{\log_2(i)} \quad (5.1)$$

$$nDCG_k^u = \frac{DCG_k^u}{IDCG_k^u} \quad (5.2)$$

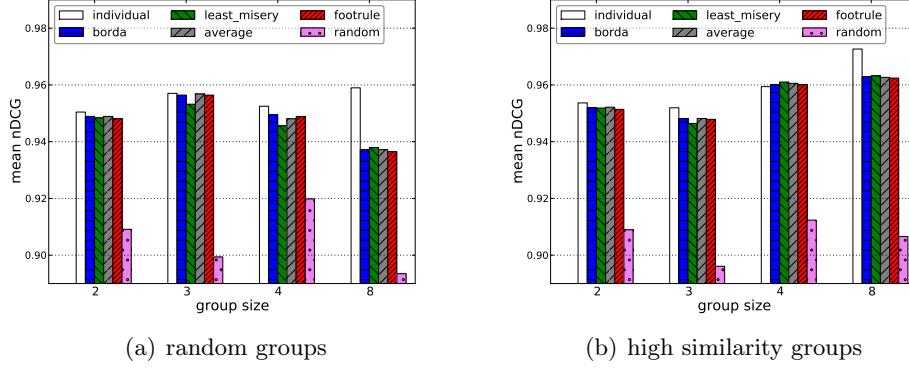


Figure 5.2: Effectiveness of group recommendation with rank aggregation techniques.

where IDCG is the maximum possible gain value for user u that is obtained with the optimal re-order of the k items in p_1, \dots, p_k .

To compute nDCG we need to know the true user rating for all the items in the recommendation list. Actually, when the test set (items rated by the users) contains only some of the items ranked in the recommendation list one must update the above definition. In our experiments we computed nDCG on all the items in the test set of the user sorted according to the ranking computed by the recommendation algorithm (individual or group recommendations). In other words, we compute nDCG on the projection of the recommendation list on the test set of the users. For example, imagine that $r = [1, 4, 5, 8, 3, 7, 6, 2, 9]$ is a ranked list of recommendations for a group. Since this is a group recommendation list, as we observed above, none of the items in this list occurs in the training set of any group member. Moreover, suppose that the user u test set consists of eight items $\{1, 4, 7, 8, 9, 12, 14, 20\}$. In such case, we would compute nDCG on the ranked list $[1, 4, 8, 7, 9]$.

5.3 Experimental Results

We start our evaluation by comparing different aggregation methods on groups of different sizes (Section 5.3.1). In Section 5.3.2 we illustrate the main contribution of this chapter, i.e., the dependency of the user satisfaction, for the group recommendations, on the user satisfaction for the individual recommendation.

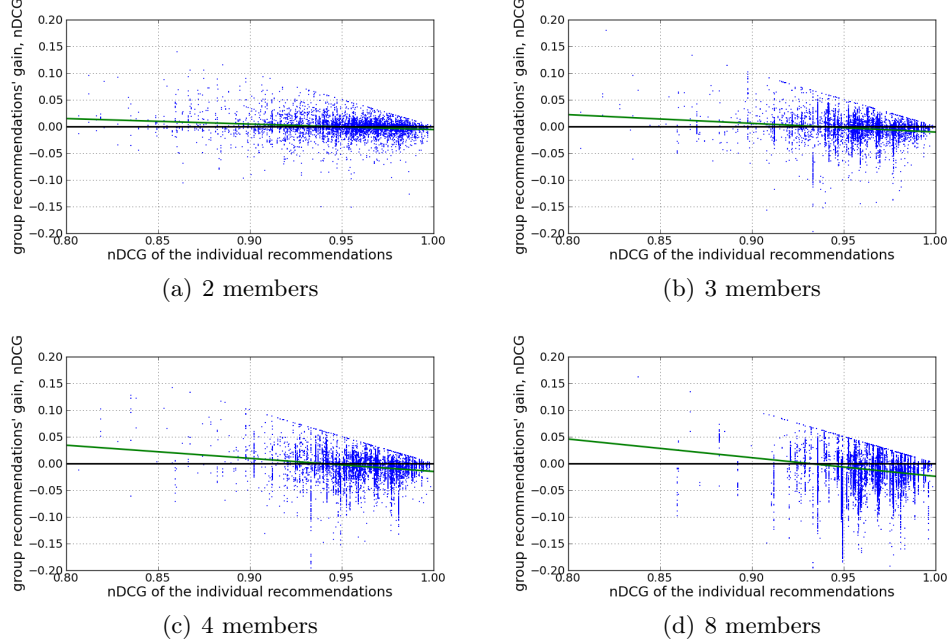


Figure 5.3: Gain of group recommendations with respect to individual recommendations.

5.3.1 Effectiveness of Group Recommendations

In the first experiment we evaluate the effectiveness of the group recommendations when varying the aggregation method and the group size. We compare that with the effectiveness of the individual recommendation. We repeated this experiment for random groups and groups with high inner group similarity. Our initial hypothesis was that the effectiveness of the group recommendation should decrease as the group size increases. In fact, our intuition says that it is harder to build good recommendations for larger groups as it gets harder to find a consensus among many, potentially different preferences. Moreover, when building recommendations for large groups we do not consider items that are already experienced (are in the training set) by any of the user in the group. This is likely to discard the most popular items, making it harder to make “everybody likes this” type of recommendations.

Figure 5.2 shows the results of our experiments using random groups and group with high inner similarity and illustrates how this intuition in some cases could be wrong. We computed the average effectiveness (nDCG), over all the users in any group, of the group recommendations built using the five presented aggregation methods for group sizes equal to 2, 3, 4 and 8. In Figure 5.2 the

effectiveness of the group recommendations is also compared with that of the individual recommendations. These results are based on a sample consisting of 1000 groups for each experimental condition (e.g., 1000 random groups of size 8).

Firstly, we observe that varying the group size the variation of the effectiveness of the group recommendations is not large for groups of size 2, 3, and 4. Moreover, we see that increasing the group size the effectiveness of the group recommendations tends to decrease only for randomly generated groups. This does not hold for groups with high inner similarity; in fact, the recommendations for groups with eight members have the largest effectiveness. We note that recommendations for random groups of size 3 deviate from the general behavior: they are more effective than those for random groups of size 2. For high similarity groups the opposite holds. We do not have an explanation for that.

Random aggregation performs uniformly worse than any other method. The difference between random and other aggregation methods varies from 3 to 6%. Such a small variation in performance could seem surprising, however, it can be explained by the following example. Continuing the example started in Subsection 5.2.3, imagine to compute nDCG for the items in the user test set: $\{1, 4, 8, 7, 9\}$. Imagine that the user has given the corresponding ratings for these items: $[5, 5, 4, 3, 2]$. If the group (or personal) recommendation returns this perfect ordering, then one has a perfect match with user preferences and $\text{nDCG} = 1.0$. However, imagine, that CF rating prediction method made some mistakes and returned the ranked list $\{1, 8, 4, 9, 7\}$, with the corresponding ratings: $[5, 4, 5, 2, 3]$. nDCG of this list is still large and it is equal to 0.97. Further, if one generates a sample of random orderings of this list, on average nDCG is 0.91; which explains the high values of random aggregation in the previous experiment.

Comparing the effectiveness of group recommendations to the individual recommendations, as expected, one can observe that the individual recommendations are better ranked than the group recommendations. But this difference is very small in the groups of size 2, 3, and 4. A noticeable difference is observed only for the largest groups, i.e., with 8 members. This shows that rank aggregation is an effective approach for building group recommendations, especially for small groups, when the objective of the recommender system is to correctly rank the recommendations and not to predict the correct rating.

These results show that combining potentially conflicting rankings in a group could create a group recommendation that is not satisfactory for the group members. This happens for large groups of size 8 in both random and highly similar groups. But, even for large groups, aggregating the ranked lists predictions of the users in a group in some cases can have a positive effect. This could happen if the individual predictions are not very good. In this situation, the combination of the ranked lists of the group members can fix errors performed by the individual predictions.

This conjecture is tested in the next Section 5.3.2 and was originated by the ob-

servation that in many of our experimental conditions the group recommendation effectiveness is not substantially inferior than the individual one.

In conclusion, we also notice that the aggregation method itself has not a big influence on the quality, except for random aggregation. Moreover, we observe that there is no clear winner and the best performing method depends on the group size and inner group similarity.

In the following sections, we analyze the correlation between the effectiveness of the group and the individual recommendations, and for this goal we focus on groups of four users. This is a common group size for many activities such as movie watching with friends or traveling.

5.3.2 Group Versus Individual Recommendations

In the second experiment we measure the difference between the effectiveness of the individual and the group recommendations' lists. We want to understand *when* the group recommendations are better or worse (ranked) than the individual recommendations. We call this difference the “gain” of effectiveness of the group recommendations. A positive gain means that the group recommendations are better ranked than the individual recommendations. As it was done in the previous experiment, we performed experiments with random groups and groups with high inner group similarity. We show results only for the average aggregation method and the high inner similarity groups. However, the results are very similar for the other settings using random groups and Borda, Spearman Footrule or Least misery aggregation methods (not shown here for lack of space).

Figure 5.3 shows a scatter plot where each user, in a group, is represented by a point. Here, the x axis measures nDCG for the user individual recommendation list, while the y axis shows the gain in nDCG of this group recommendation list for the same user. Note that the same user can be in several different groups, hence a user may be represented by several points. We plot only data for a sample of 3000 groups. We can see that there is a negative correlation between the gain and the effectiveness of the individual recommendations, i.e., the gain decreases as the effectiveness of the individual recommendations increases. This means that the worse (ordered) the individual recommendations are, the better (ordered) the group recommendations are, i.e., the more valuable is to aggregate the recommendation lists of the group members.

We visualized this tendency by plotting the best fit lines through the points in the scatter plots. The correlation of the fit is significant ($p < 0.001$). Specifically, it is -0.18 for groups of size 2, -0.17 of size 3, -0.19 of size 4, -0.18 of size 8 for groups with high inner group similarity and average aggregation method. We also observe that the best fit line is getting steeper when group sizes increase.

From the analysis of this experiment we can conclude that the worse are the individual recommendations the better are the group recommendations built

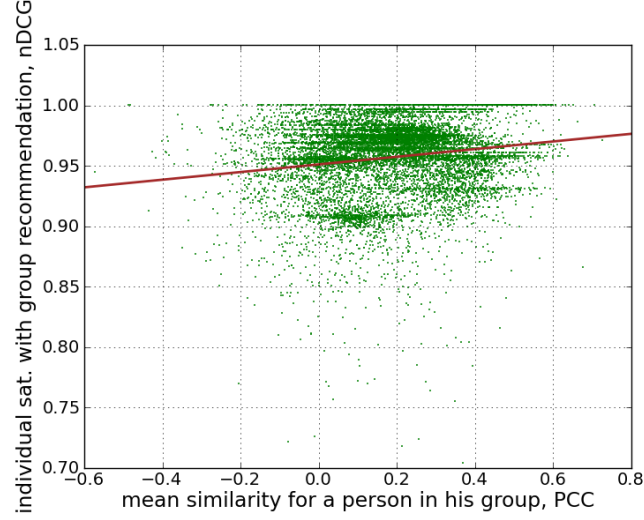


Figure 5.4: Correlation of the effectiveness of group recommendations with the average similarity of the user with the other group members.

aggregating the recommendations for the users in the group. This is an interesting result as it shows that in real RSs, which always make some errors, some users may be better served with the group recommendations than with recommendations personalized for the individual user. When the algorithm cannot make accurate recommendations personalized for a single user it could be worth recommending a ranked list of items that is generated using aggregated information from similar users.

This result motivated the following experiment where we look at the correlation between the average similarity of a user to the other group members and the effectiveness of the group recommendations for this user. In other words we measured the effectiveness of the group recommendations for a user as a function of the user's similarity to their peers. Here we use random groups since they better sample the possible user-to-user similarities that can be observed in groups.

Figure 5.4 shows the results of this experiment together with the best fit line. On the x axis we plot the average similarity of the user to all the other members of his group and on the y axis the effectiveness of the group recommendation for that user. First of all, we notice that there is a positive $r=0.22$ (statistically significant $p<0.001$) correlation between the similarity to the other group members and the user satisfaction. This shows, that when a user is more similar to the members of the group the group recommendation is better ordered. We observe that this correlation is not as strong as we expected; we believe this topic deserves further analysis.

5.4 Discussion and Conclusions

In this chapter we analyzed the effectiveness of ranked list recommendations tailored for a group of users. We propose to use rank aggregation techniques for generating group recommendations. We show that these recommendations are only slightly less effective than the individual recommendations for groups of moderate size (2, 3, and 4). Only for larger groups, of size 8, the group recommendation is significantly inferior than the individual recommendation.

Moreover, our results revealed some novel and non-obvious facts. First, we observed that the effectiveness of group recommendations does not necessarily decrease when the group size grows. In fact, this is not happening for groups of similar users. Secondly, when the individual recommendations are not correctly ranked, then recommending items ordered for a group recommendation can improve the effectiveness of the recommendations. This indicates that when, for some reason, the individual recommendations are not good, aggregating the ranked list recommendations built for a group of users, which the target user belongs to, can increase recommendation goodness.

Third, we have experimentally confirmed a common belief that the more alike are the users in the group, the more satisfied they are with the group recommendations.

In the future, we want to test our findings with other data sets, with other recommendation algorithms, and with real users. In such a way we want to further validate our findings, which are currently based on synthetically generated groups and may also be dependent on the particular collaborative filtering technique that we used, namely SVD.

In particular we would like to assess the impact in our group recommendation techniques of collaborative filtering prediction approaches aimed at producing better rankings [Weimer et al., 2007] rather more accurate rating predictions. We also want to further investigate when it is convenient to make a group recommendation instead of an individual recommendation. We consider this approach as a semi-personalized recommendation strategy. With that respect it would be important to find critical user or group features that can help to decide when it is beneficial to make an individual, or a group, or even a non-personalized recommendation.

Item Weighting and Item Selection

A large part of Collaborative Filtering (CF) recommender systems generate rating predictions for a target user by exploiting the ratings of similar users [Resnick and Varian, 1997, Herlocker et al., 1999]. Therefore, the computation of the user-to-user similarity is an important element in CF; it is used in the neighborhood formation and rating prediction steps. In this chapter we investigate the role of item weighting and item selection techniques. These allow to develop an adaptive version of the user-to-user similarity, i.e., they dynamically change the computation depending on additional information at hand, such as the profiles of the compared users, and the target item whose rating prediction is sought.

We will show that items can be weighted according to their importance for predicting the rating of other items. This weight can be computed as the correlation between two items' rating vectors: the predictive and the predicted ones. In this chapter we analyze a wide range of item weighting schemas. Moreover, we introduce two item selection approaches. They compute the similarity between two users using only a subset of co-rated items which best describes the taste of the users with respect to the target item. These are the items which have the highest correlation with the target item.

Item weighting and item selection techniques could be extended to incorporate contextual information into the memory based CF approaches. Imagine a Recommender System in the tourism domain. Here, the context can be used to understand that some items should be more likely to be recommended than others, and also that some item ratings are more relevant than others as it was assumed in [Adomavicius et al., 2005]. For example, an open air museum is worth recommending only on a sunny day. When predicting the rating for the target item we propose to use only the rating data of the most similar items. The rationale is that context should influence the selection of the target item and the items used to generate the prediction, since their ratings should provide

situated knowledge. For example, if we are predicting the rating of an open air museum on a sunny day, we use only the ratings of similar points of interest, i.e., open air theaters or other open air museums, and these points of interest are all worth visiting on a sunny day. In this sense, the target item plays the role of the context descriptor and the predictive items are dynamically selected with respect to it. The similarity between items may be computed using descriptive features of items (i.e., open air attraction) or analyzing the mutual dependencies between their ratings.

Another way to incorporate context into the user-to-user similarity function would be to use rating weighting and selection techniques instead of item weighting and selection. This would follow a line of works initiated by a first attempt to develop a CACF [Chen, 2005] (also mentioned in [Adomavicius and Tuzhilin, 2010]). Chen extended user-to-user CF by incorporating contextual information into the similarity computation step. The author proposed to use Pearson Correlation Coefficient to compute the similarity of two contexts based on the ratings that were given in these contexts. This similarity is later used to determine the rating weight - the more similar was the context to the target context, the more influential is the rating. Note, that to our best knowledge this approach was never evaluated.

We stress here that in this chapter we are focusing on user-to-user CF, as opposed to item-to-item CF. The latter computes the rating prediction for a target user and item pair as a weighted average of the ratings the user gave to *similar items*, and therefore user-to-user similarity is not exploited. In user based CF, user-to-user similarity computation is used in the neighborhood formation and in the final rating prediction. Two users are considered similar if they have expressed similar opinions (ratings) on a set of co-rated items. In standard CF each of the co-rated items contributes equally to the similarity. But it seems natural to conjecture that some items could be more important than others when computing the user-to-user similarity. For example, if the movie “Dead on Arrival” gets the lowest possible rating from all the users, then it should not be very useful in understanding whether two users have similar preferences. These type of items, i.e., with very similar ratings in the users’ population, should contribute less (in case of item weighting) than others or could be even discarded (in case of item selection) from the similarity computation.

Moreover, additional information about the *content* of the items could also be exploited to improve the performance of CF. For example, consider two users who have very similar tastes for action movies but very different tastes for dramas and documentary movies. In general, these users would have a small correlation when comparing all their co-rated items. However, when considering only the action movies the correlation would be higher. When predicting a rating for an action movie, co-rated action movies could be given a higher importance (weight). Incorporating such additional information is not a trivial task. There are many

sources of information potentially useful for computing the importance of the item. Moreover, it is not clear how this information can be integrated into the CF prediction model. This is particularly complex for item selection, as the rating matrix is sparse. This makes classical feature selection methods ineffective. In addition, it is also unclear whether to compute the item importance it would be better to consider the rating data statistics, or to exploit information coming from external sources, such as the item descriptors or other contextual information. We also need to determine how strongly the additional information should influence the similarity. Finally, we observe that after having found good information sources (for determining the importance of items), one still has to cope with the problem of adapting the user-to-user similarity definition. In fact, there is no well-accepted way (formula) to extend the standard user-to-user correlation measure in order to take into account these weights.

In this chapter we address the issues mentioned above, and in particular:

- We analyze a wide range of item weighting schemas — some new and some already used in previous research. These are based on various types of information; one group of methods explores the dependencies between item ratings, whereas another tries to exploit the information contained in the descriptions of the items.
- We empirically investigate three approaches to incorporate the computed weights into the user-to-user correlation measure and we provide an analysis of their behavior. We also conduct some experiments aimed at understanding to what extent the additional information contained in the weights should modify the prediction formula, compared to the standard similarity metric.
- We analyze in particular the behavior of an item selection method based on the computed weights. This study evaluates whether it is possible to improve the prediction accuracy by using just a small subset of items, i.e., those that are very relevant to the target item.

We show that item weighting can improve the accuracy of the baseline CF (for both datasets), for all the neighborhood sizes, and for all the error measures we used. This improvement is obtained by a particular weighting method based on Singular Value Decomposition (SVD-PCC) of the rating matrix. The other weighting schemas did not uniformly improve the baseline. However, in some situations they can even produce larger improvements over the baseline than SVD-PCC. We also show that the largest reduction of the mean absolute error was achieved by using item selection together with a particular weighting method but at the cost of sacrificing coverage. We also identified a weight incorporation method that uniformly performs better than the others.

6.1 Adaptive Similarity

In this section we show how to compute the item weights using a weighting schema, and then how to incorporate these weights into the standard user-to-user similarity measure. To compute these weights we use a wide range of schemas that are described in Subsection 6.1.1. Each weighting schema computes a real valued $n \times n$ weight matrix $W_{n \times n}$ (n is the number of items). The entry w_{ij} is the weight (importance) of item i for predicting the ratings of item j (for all the target users). In order to be able to compare the behavior of different weighting schemas, in this chapter we normalized the weights to range in $[0, 1]$.

The role of the target item j needs some further explanation. In fact this gives to an item weighting schema the flexibility to assign to each predictive item a weight that depends on the target item whose rating predictions are sought. Hence, the weights used for predicting the ratings for item j (for all the users) can be different from those used in the predictions for another item j' . In this way, we encode in a weight how much two items are correlated, and what role an item should play when a prediction is sought for the second one. Without such a flexibility we could only express the general knowledge about the importance of an item for all the rating predictions. In fact, an example of this approach is provided by the variance weighting method that is explained in Subsection 6.1.1. Finally, given a weighting schema, there are several ways to integrate the weights into an (unweighted) similarity measure. Methods for item weighting are described in Subsection 6.1.2 and methods for item selection in Subsection 6.1.3.

6.1.1 Weighting Schemas

Item weights tries to estimate how much a particular item is important for computing the rating predictions for a target item. In CF, item weights can be learned while exploring training data consisting of user ratings or using additional information associated with the items, i.e., their descriptions. Based on this observation we can partition weighting methods in two groups. The methods in the first group determine the item weights using statistical approaches and are aimed at estimating statistical properties of the ratings and the dependencies between items. For instance, the variance of the item ratings (provided by a users' population) belongs to the first group of methods. The second group of methods uses item descriptions to estimate item dependencies and finally infer the weights. For instance, the similarity between item descriptions can be used to infer item-to-item dependencies and these can be converted into weights.

In the rest of this chapter we will consider the following item weighting approaches: Random, Variance, Mutual Information, Genre, Pearson Correlation Coefficient (PCC), and PCC computed on low dimensional item representations.

Let us now precisely define these methods.

Random. The first method is used as a reference for comparing different approaches. It assigns to each predictive (i) and target (j) item combination a random weight sampled from the uniform distribution in $[0, 1]$: $w_{ij} = \text{random}([0, 1])$

Variance. The variance method is based on an observation originally made in [Herlocker et al., 1999]. It gives higher weights to items with higher variance among the ratings provided by the users to that item:

$$w_{ij} = \frac{\sum_u (r_{ui} - \bar{r}_i)^2}{\#users \text{ who rated } i}$$

Here the sum runs over all the users who rated the item i , r_{ui} is the rating given by user u to item i , and \bar{r}_i is the mean of the ratings provided by all the users to the item i . The variance feature weighting method uses only information about the predictive item and the weight does not depend on the target item. Conversely, all the methods that we are presenting next, explore the relations between predictive items and the target item.

IPCC. The first method in this group uses Pearson Correlation Coefficients (PCC) as a measure of the dependency between two vectors representing the ratings of all users for two items:

$$w_{ij} = \frac{\sum_u (r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_u (r_{ui} - \bar{r}_i)^2 \sum_u (r_{uj} - \bar{r}_j)^2}}$$

Here the index u runs over all the users that have rated both items i and j , and \bar{r}_i is the mean of the ratings on item i . We observe that IPCC (Item Pearson Correlation Coefficients) builds a symmetric weight matrix W , i.e., the importance of the item i in the prediction of the ratings for the item j is equal to the importance of the item j when it is used to predict the ratings for the item i .

Mutual Information. Mutual Information (MI) measures the information that a random variable provides to the knowledge of another. In CF, following [Yu et al., 2003], we compute the mutual dependency between the target item variable and the predictive item variable (the values are the ratings). The Mutual Information of two random variables X and Y is defined as:

$$I(X; Y) = \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$$

The above equation can be transformed into $I(X; Y) = H(X) + H(Y) - H(X, Y)$, where $H(X)$ denotes the entropy of X , and $H(X, Y)$ is the joint entropy of X and Y . Using the above formula Mutual Information computes the weights as follow:

$$\begin{aligned}
w_{ij} = & - \sum_{v=1}^5 p(r_i = v) \log p(r_i = v) - \sum_{v=1}^5 p(r_j = v) \log p(r_j = v) \\
& + \sum_{v'=1}^5 \sum_{v''=1}^5 p(r_i = v', r_j = v'') \log p(r_i = v', r_j = v'')
\end{aligned}$$

Here the probabilities are estimated with frequency counts in the training dataset, hence for instance $p(r_i = v) = \frac{\text{\#users who rated } i \text{ as } v}{\text{\#users who rated } i}$ estimates the probability that the item i is rated with value v . We observe that v is running through all rating values, and in our data sets this is equal to 5.

SVD-PCC. The two methods described previously, i.e., IPCC and Mutual Information, compute a form of statistical dependency between two items. Note that such statistics between two items are computed using the pairs of ratings that the users expressed for *both* items. When the data is extremely sparse there can be only few users who have rated both items, and the computation of the statistical dependency based on this incomplete information could be inaccurate. Moreover, the fact that two items were not co-rated by the users should not imply that they are not similar [Billsus and Pazzani, 1998]. In order to avoid such problems we decided to reduce the dimensionality of the ratings matrix and compute item-to-item correlation in the transformed space rather than on the original raw data. This approach is based on Latent Semantic Analysis (LSA) [Deerwester et al., 1990], originally used for analyzing the relationships between a set of documents and the contained terms. LSA is now widely applied to CF to generate accurate rating predictions, and the user-to-user similarity is computed in a space with lower dimension than the number of items and users [Sarwar et al., 2000]. LSA uses Singular Value Decomposition (SVD), a well-known matrix factorization technique (see [Golub and Van Loan, 1996] for a reference).

SVD factors a rating matrix $R = (r_{ij})$, of size $m \times n$ (m users and n items) into three matrixes as follows: $R_{m \times n} = U_{m \times rk(R)} S_{rk(R) \times rk(R)} V_{n \times rk(R)}^T$ where, U and V are two orthogonal matrices, $rk(R)$ is the rank of the matrix R and S is a diagonal matrix having all the singular values of matrix R as its diagonal entries. Note, that we used 0 as a missing rating. All the singular values are positive and ordered in descending order. Selecting the $d < rk(R)$ largest singular values, and their corresponding singular vectors U and V , one gets the rank d approximation $F_{m \times n} = U_{m \times d} S_{d \times d} V_{n \times d}^T$ of the rating matrix R .

Each row of the matrix $V_{n \times d}$ gives a dense representation of the corresponding item in a d dimensional space. The choice of d is an open issue in the factor analytic literature [Deerwester et al., 1990] and it is typically application dependent. We set $d = 40$ in our experiments (see Section 6.2 for more details). To compute the

similarity of item i and item j we compute the PCC between two columns of V :

$$w_{ij} = \frac{\sum_d (V_{id} - \bar{V}_i)(V_{jd} - \bar{V}_j)}{\sqrt{\sum_d (V_{id} - \bar{V}_i)^2 \sum_d (V_{jd} - \bar{V}_j)^2}}$$

where the sum runs over the row index of the matrix V and \bar{V}_j denotes the average value of the j -th column.

Genre Weighting. All the previous methods exploit statistics of the users' rating data to determine item weights. The last method that we present here computes the weights using descriptions of the items. In the movie recommendation data sets, which we used for our experiments, the movies are tagged with movie genres. Hence, we make the assumption that the larger the number of common genres, the higher is the dependency. Hence, the weight of the predictive item i for predicting the ratings of the target item j is given by:

$$w_{ij} = \frac{\# \text{ common genres of items } i \text{ and } j}{\# \text{ genres}}$$

Genre weighting is related to the methods presented in [Berkovsky et al., 2007], where item description information is used to selectively choose the items to be used in the user-to-user similarity.

6.1.2 Weight Incorporation

The two most popular measures of user-to-user similarity are: the cosine of the angle formed by the rating vectors of the two users; and Pearson Correlation Coefficient (PCC). PCC is preferred when data contains only positive ratings, and has been shown to produce better results in such cases [Breese et al., 1998]. The PCC between the users u and v is defined as:

$$PCC(u, v) = \frac{\sum_i (r_{ui} - \hat{r}_u)(r_{vi} - \hat{r}_v)}{\sqrt{\sum_i (r_{ui} - \hat{r}_u)^2 \sum_i (r_{vi} - \hat{r}_v)^2}}$$

where r_{ui} denote the rating given by the user u to the item i , and \hat{r}_u is the mean of the ratings of the user u . The sum runs over all the items i that both users have rated.

As we mentioned in the Introduction, the motivation for using weights in the user-to-user similarity is to improve the prediction accuracy. This approach has been also used in instance-based learning to produce a weighted version of the Euclidean metric [Aha, 1998]. The weighted Euclidean distance between two d -dimensional vectors x and y with weight vector z is defined as:

$$d_w(x, y) = \sqrt{\sum_{j=1}^d z_j (x_j - y_j)^2}$$

We must observe that the effect of the weights on the Euclidean metric has a clear (geometric) interpretation: the features with larger weights produce a greater impact on the similarity. For PCC the situation is not as clear. In addition, because of the more complicated nature of PCC, there exists no single well accepted approach to extend it with weights. In our study we analyzed three different versions of *weighted* PCC. The first version which we call normalized Weighted PCC ($WPCC_{norm}$) makes a natural extension of the unweighted PCC:

$$WPCC_{norm}(u, v, j) = \frac{\sum_i w_{ij}(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_i w_{ij}(r_{ui} - \bar{r}_u)^2 \sum_i w_{ij}(r_{vi} - \bar{r}_v)^2}}$$

where j denotes the target item of the rating prediction, and w_{ij} is the weight of the (predictive) item i when making a prediction for j . Here \bar{r}_u is the average rating of the user u .

The second approach we consider is used by SASTM[SAS, 1999] and it extends the previous basic approach by replacing the standard average of the user ratings by a weight-normalized version:

$$WPCC_{wavg}(u, v, j) = \frac{\sum_i w_{ij}(r_{ui} - \hat{r}_u)(r_{vi} - \hat{r}_v)}{\sqrt{\sum_i w_{ij}(r_{ui} - \hat{r}_u)^2 \sum_i w_{ij}(r_{vi} - \hat{r}_v)^2}}$$

where $\hat{r}_{ui} = \frac{\sum_i w_{ij} r_{ui}}{\sum_i w_{ij}}$ and we note that this new normalized average depends on the target item j . The idea here is that the *weighted* average of the user ratings should better estimate the user average. Note that we compute the average rating using all the ratings of the user, rather than just overlapping items.

The third method that we consider in this chapter was used in [Jin et al., 2004], and it differs from the previous proposal as it does not normalize the similarity metric:

$$WPCC_{no-norm}(u, v, j) = \frac{\sum_i w_{ij}(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_i (r_{ui} - \bar{r}_u)^2 \sum_i (r_{vi} - \bar{r}_v)^2}}$$

In this approach, the users who co-rated highly weighted items would have higher correlation. Because of the lack of the normalization this correlation measure depends on the absolute values of weights, rather than on the relative differences between the weights. Therefore, the similarity that is computed on the items with small weights will be small.

Further discussion on the different weight incorporation methods can be found in the experimental part of this chapter.

6.1.3 Item Selection

In item selection instead of using the precise values of the weights for tuning the similarity function, a simpler decision is taken to either use or not an item, based

on the value of its weight. In fact, item selection could be seen as a particular case of item weighting, where just binary weights are used.

Extremely sparse rating matrix makes classical filter method ineffective. Imagine for instance, that there are two users that are perfectly correlated but have co-rated just a few items and these items have small weights. One of the two users could be used to predict the ratings of the second user. But, if we straightforwardly select a small number of items, according to a precomputed weight (i.e., just the items with largest weights), there is a very small chance that the profiles of these users will overlap on the selected items. Therefore, using this simple method many good neighbors could be discarded hence decreasing the prediction quality.

Moreover, finding the optimal subset of items would require to conduct an extensive search in the space of all the subsets of the items [Kohavi and John, 1997]. Applying this to a recommender system scenario would require to conduct a search procedure for every target item (the item playing the role of the class to be predicted) and for a large number of subsets of the predictive items. This is clearly extremely expensive, and therefore, we propose to use a more parsimonious approach (filter method) that uses information provided by a item weighting method to select, for each target item and user pair, an appropriate set of predictive items.

For these reasons, we propose a localized item selection method called **BIPO** (Best Items per Overlap). BIPO selects the subset of items with the largest weights from the set of the items co-rated by both users whose similarity we want to determine. We shall explain BIPO method using a simple example of user-item rating matrix showed below:

w_{ti}	i_1	i_2	i_3	i_4	i_5	i_t
u_1	5	3	2	1	?	6
u_2	4	2	4	?	5	?

The table consists of two users and six items. The question marks indicate the unknown ratings. Let us assume that we want to predict user's u_2 rating for the item i_t . Moreover, suppose that we have computed item weights beforehand, using an item weighting algorithm. The weights are showed in the second row of the table. For example, the weight of item i_1 for predicting the target item i_t is $w_{t1} = 0.1$.

In the example above, BIPO method would select items with the highest weights that are rated by both users. Suppose we want to compute user-to-user similarity on the 2 most largely correlated items. In such a case BIPO would select items 3 and 2, and not 4 or 5, despite the fact that these have a larger correlation with the target item. We note that in BIPO the items used in the prediction change for every target item and neighbor user pairs.

In this chapter we also propose and evaluate another item selection approach. Instead of performing a dynamic item selection which depends on the target user we simply filter out (and never consider in the similarity computation) the items with the smallest weights. In other words, if a weight is less than a given threshold, we set it to zero. Note, that instead of using binary weights as we did in BIPO, we still use in the user-to-user similarity computation the weights in $[0, 1]$. Moreover, the set of items used in a prediction depends only on the target item and it is the same for all the users.

6.2 Experimental Evaluation

In this section we evaluate the item weighting and item selection methods presented so far. We observe that to generate the predictions, one of the three variations of *WPCC* was used in computing the nearest neighbors of the target user and also in the prediction step:

$$r_{ui}^* = \bar{r}_u + \frac{\sum_{v \in N(k, u, i)} WPCC(u, v, i) \times (r_{vi} - \bar{r}_v)}{\sum_{v \in N(k, u, i)} |WPCC(u, v, i)|}$$

Here the sum runs over the k -nearest neighbors $N(k, u, i)$ of the user u . Note that the neighbors depend on the target item i because the user-to-user similarity depends on the target item. In our implementation of CF, as previously done in other studies, when making a rating prediction for a user u we take into account only the neighbors with at least a minimum number of co-rated items with the target user (six in our case) [Berkovsky et al., 2007].

In the experiments, we used two data sets with ratings in $\{1, 2, 3, 4, 5\}$. The MovieLens [MovieLens, 2007] data set contains 100K ratings, for 1682 movies by 943 users, who have rated 20 and more items. The data sparsity of this dataset is 96%. The Yahoo! [Yahoo, 2007] Webscope movies data set contains 221K ratings, for 11915 movies by 7642 users. See Appendix C for the details on the data sets. The data sparsity of this dataset is much higher, 99.7%. To evaluate the described methods we used holdout validation, where both datasets were divided into train (80%) and test (20%) subsets. We used the train data to learn the weights and also to generate the prediction for the test ratings. Because of the high computational complexity, when computing weights, we were not able to perform a multi-fold cross-validation.

To measure the accuracy we used Mean Absolute Error (MAE), coverage, F_1 (in the rest of the chapter denoted as F), High MAE, precision and recall [Herlocker et al., 1999, Herlocker et al., 2004]. To compute F , precision and recall, we considered items worth recommending (relevant items) only if their ratings were 4 or 5. Since, we are interested in recommending only the top items, we propose to modify the MAE error measure to see how an algorithm performs

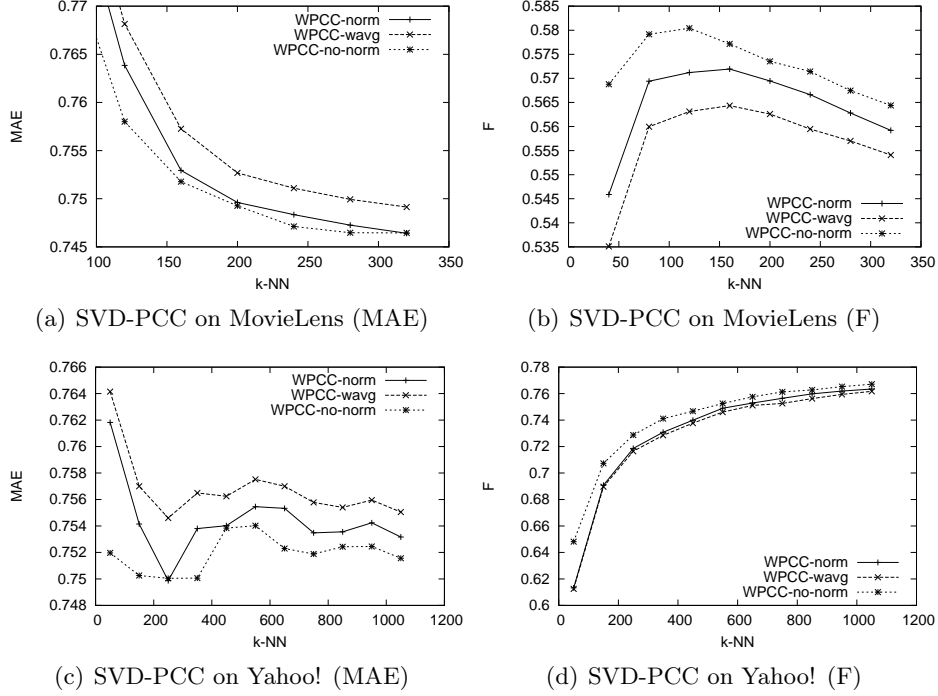


Figure 6.1: Performance of weighted PCC measures.

on the predictions of items with highest ratings. For this purpose we defined High MAE measure as MAE computed only on the items that were rated by the user 4 or 5. Coverage is the fraction of the ratings in the test set for which predictions can be made. Note that when computing MAE we followed the standard practice and did not take into account the ratings that an algorithm was not able to predict.

To compute Singular Value Decomposition we used the SVDLIB [svdlib, 2008] library and selected $d = 40$ biggest singular values. As mentioned earlier, the selection of d is a problematic issue. As stated above, in our experimental setting the parameter selection using cross-validation was too expensive. Selecting a too small d can lead to a big loss of information; however, selecting too big d could result in not removing the noise present in the data. Computing similarity in the higher dimensional space might also not be effective because of the “curse of dimensionality”. Our selection was based on some trials and the analysis of the outcome of previous reports [Deerwester et al., 1990, Billsus and Pazzani, 1998].

6.2.1 Evaluation of Weight Incorporation

We started our experiments by evaluating the performance of the three weight incorporation methods described in Subsection 6.1.2. In a first experiment we have used all the weighting schemas and computed F and MAE for the three weight incorporation methods. The comparison of the weight incorporation methods introduced in Section 6.1.2 for the two considered datasets, using the *SVD-PCC* weighting schema, is depicted in Figure 6.1.

We observe here that most of the time $WPCC_{no-norm}$ performs better than the other methods independently of the data set, and independently of the error measure we used. Because of lack of space we do not show the results obtained using the other weighting schemas, but we summarize them saying that they confirm what has been observed for SVD-PCC, i.e., $WPCC_{no-norm}$ it is always the best performing approach for using the weights into the user-to-user similarity metric.

This is quite surprising, as $WPCC_{norm}$ or $WPCC_{WAVG}$ are used in the majority of the literature and software packages [SAS, 1999, Yu et al., 2003]. In fact, as we mentioned above, because of the complicated behavior of PCC these results are hard to interpret. Our explanation is that $WPCC_{no-norm}$ gives a higher absolute correlation to a user who expressed ratings on more correlated items to a target item. For example, consider the following three users with ratings:

w_{ti}	0.8	0.7	0.6	0.5
	i_1	i_2	i_3	i_4
u_1	?	?	1	5
u_2	2	4	2	4
u_3	1	5	?	?

Here the first line of the table gives the item weights which are used in the similarity computation. Computing the user-to-user similarity on this simple example gives an insight into the nature of the different correlations. We have the following: $WPCC_{norm}(u_1, u_2) = 1$, $WPCC_{norm}(u_3, u_2) = 1$, $WPCC_{no-norm}(u_1, u_2) = 0.55$, $WPCC_{no-norm}(u_3, u_2) = 0.75$. One can see that $WPCC_{norm}$ takes into account only the relative size of the weights that are used in the similarity computation and not their absolute value. In fact, the similarity of u_1 and u_2 is equal to the similarity of u_3 and u_2 , even if the weights used for u_1 and u_2 are smaller. Therefore, computing the user-to-user similarity on highly correlated items (to a target item) or weakly correlated items does not change the final similarity and it remains equal to 1. On the contrary, $WPCC_{no-norm}$ takes into account the absolute values of the weights. This leads to a higher user-to-user correlation if computed on the item ratings which have bigger weights, i.e., are more correlated with the target item. Therefore, the users whose correlation is computed on the items with small weights will likely not to be in the target user neighborhood.

6.2.2 Evaluating Weighting Schemas

The second experiment evaluates all the weighting schemas described in Subsection 6.1.1 using $WPCC_{no-norm}$ as weights incorporation method. In Figure 6.2 the performance of all these approaches varying the number of nearest neighbors are shown for the two considered data sets. Here, the baseline method uses standard (unweighted) PCC. The performance is shown while varying the number of nearest neighbors.

It can be seen that SVD-PCC improves the MAE and F of the baseline prediction for both data sets and for all the neighborhood sizes. The situation with other weighting schemas is not so clear. For example, IPCC can reduce MAE, especially on the Yahoo! data set, and sometimes outperforms the SVD-PCC schema. The improvements of IPCC over the baseline method are smaller for the MovieLens dataset (considering MAE); however, this is achieved for all the neighborhood sizes. Surprisingly, the situation is different for F measure. While SVD-PCC still outperforms the baseline method, IPCC performs similarly to or even worse than the baseline. This is an interesting observation since IPCC is similar to SVD-PCC; the only difference is that SVD-PCC computes the correlation on a lower dimension item representation rather than on the original raw data.

Another interesting behavior can be observed for the weighting schema based on Mutual Information for the MovieLens dataset (called “Imutual” in all the Figures). This method performs similarly to the baseline with respect to MAE; however it gets a considerable improvement of F measure. After investigating the method further, we concluded that the increase of the F measure is due to a *large* increase of recall (up to 10%) at the cost of a small decrease in precision. This big improvement of recall is not clear and requires further investigation. Moreover, Mutual Information performs similarly to the baseline method for the Yahoo! data set. This result is different from the one reported in [Yu et al., 2003] where authors observed a big reduction of MAE. Note that in [Yu et al., 2003] authors used EachMovie data set and trained the weights on a small random subset of the users. We guess that the differences could also be explained by the unstable behavior of Mutual Information weighting method. Variance weighting in general does not improve baseline CF, which confirms the results of [Herlocker et al., 1999]. This method improves F measure, but it also increases MAE error. As expected, the random weighting schema always decreases the performance of the baseline CF.

Another interesting result can be observed for the genre weighting schema. In the MovieLens dataset it performs even worse than random item weighting and in fact it is the worst performing method. On the contrary, in the Yahoo! data set we have observed a significant reduction of MAE compared to the baseline

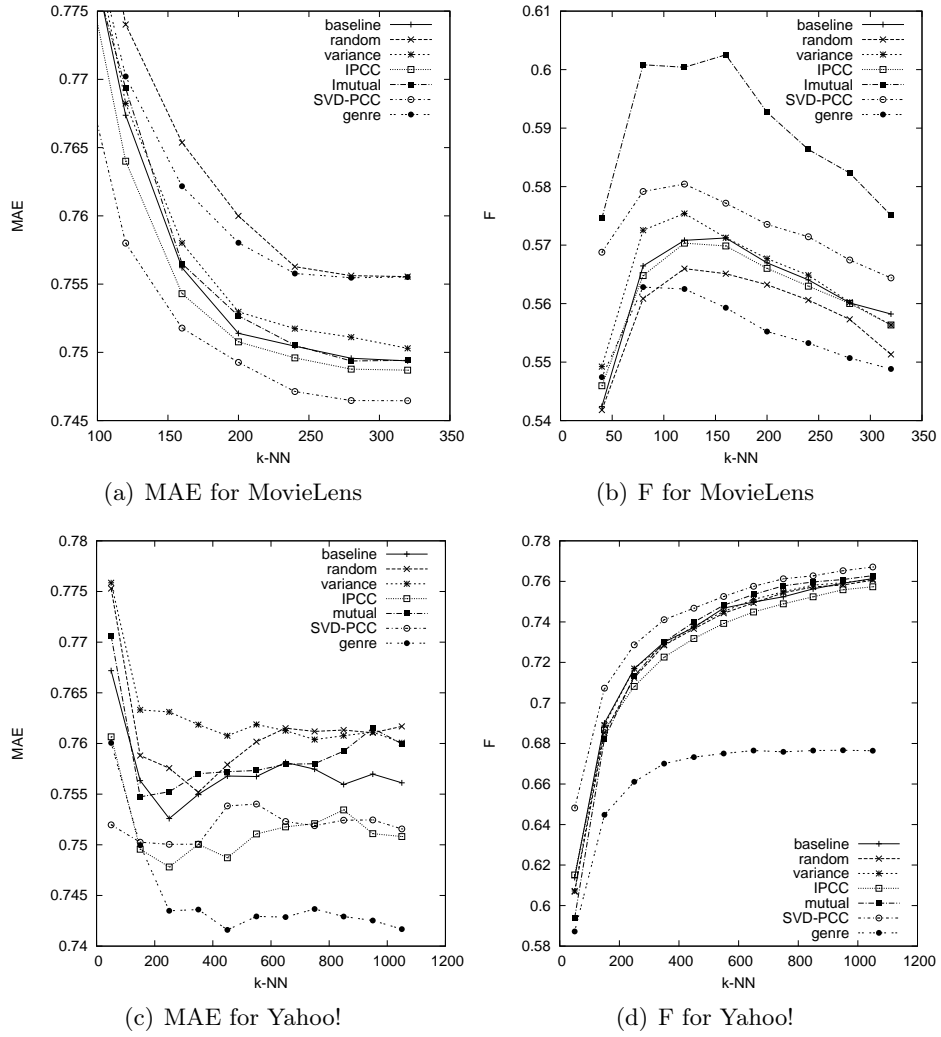


Figure 6.2: Performance of Item weighting using $WPCC_{no-norm}$ on two data sets.

method. It can be explained by analyzing the way the genre weights are computed. It is often the case that two movies do not share a single genre. Therefore, the genre overlap is 0, making the normalized weights also equal to 0. Hence, when using the genre weighting schema we perform a lot of item filtering together with item weighting.

6.2.3 Evaluating Item Selection

In this Subsection we present the evaluation of BIPO item selection method for neighbor selection. As we mentioned above, our method is computing first the item weights and later is using them to select items considered in the user-to-user similarity. We note that PCC between users *with BIPO item selection* is used while computing the neighborhood of the active user, whereas *the standard PCC, without item selection*, is used to compute the predicted rating. This is because in this chapter we want to measure the effect of item selection in neighbor formation, and consequently in CF performance.

In Figure 6.3 the performance of BIPO with all the weighting methods is depicted. We note that **all** item weighting methods used for BIPO item selection showed a better performance over the baseline CF ,i.e., collaborative filtering without item selection, for all the error measures used. This result is important since it clearly shows the robustness of item selection, and clearly shows that the improvements are due to item selection rather than weighting.

In fact, there is no single best item weighting approach and the winner depends on the particular error measure used. For example, weighting based on Mutual Information (in fig. “mutual”) produces an item selection that performs best for High-MAE error measure, whereas IPCC weighting performs better for recall measure and gives improvement up to 6.6%. Random item selection and genre labeling methods are the worst; however, they also improve the performance of the baseline method. Note that the Variance based method performs as good as IPCC and Mutual Information with respect to precision, MAE, and High MAE. This is important, because in the Variance approach the weight of the item does not depend on the target item, hence this method can be easily applied for large data sets with many items. It can be also efficiently computed and cached.

Similar performances can be seen using other datasets. The method was also tested (not shown here) on Yahoo! WebScope dataset [Yahoo, 2007], which contains extremely sparse data. We discovered that the improvements are smaller; however, they increase when the average overlap between users increases (achieved by filtering out users with small number of ratings). Our conclusion is that in order to make a meaningful item selection we must have enough co-rated items to choose from.

In the previous experiments we showed that BIPO item selection method

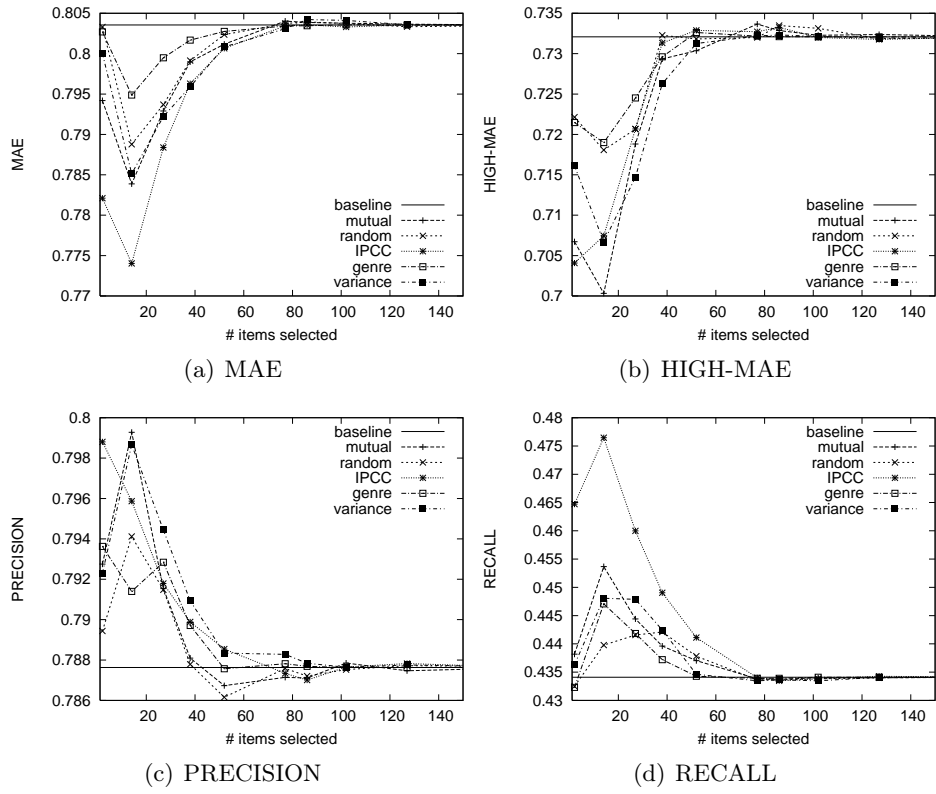


Figure 6.3: Performance of BIPO item selection methods for different error measures.

Table 6.1: Performance of BIPO and standard CF on different set of ratings.

	common ratings		CF-only ratings	BIPO-only ratings
Measure	CF	BIPO	CF	BIPO
# rat. predicted	17833	17833	117	1146
MAE	0.8024	0.7663	0.9656	0.8939
HIGH-MAE	0.7305	0.6954	1.1055	0.9776

gives good results using a small number of items in user-to-user similarity. In the next analysis we investigated whether BIPO can generate predictions for the same user-item pairs that standard CF will predict¹. Hence we evaluated the performance of BIPO on the subset of the items whose rating is predicted by both BIPO and CF, and items that were predicted only by one of the two methods.

The performance of BIPO item selection method with a fixed number (14) of selected items was compared against the baseline CF. Both methods used 60 k-nearest neighbors. The test set was divided into three subsets: “common” subset contains ratings predicted by both methods; *CF – only* contains the ratings predicted only by baseline CF and *BIPO – only* contains those predicted only by BIPO. In table 6.1 summary of the results is shown.

These results show that both methods can predict a large common subset user-item pairs in the test data set, namely 17833. On this set of user-item pairs that both can predict, BIPO decreases MAE by 4.5%. Moreover, BIPO can make more predictions compared to the baseline method (117 vs 1146 items not in the common subset). This result is surprising and shows that despite data sparsity, carefully selecting items for neighborhood formation increases recall and could also improve recommendation diversity.

The increase in recall (and also in the ability to make more predictions) shown by BIPO can be explained by better analyzing the way user-to-user similarity is computed by this method. First of all, BIPO computes the similarity using only the items that have the highest correlation with the target item. This can explain why it is more likely that the target item is also rated by the neighbors found by this method, and therefore the collaborative filtering prediction rule can actually compute a prediction for it.

Secondly, by computing the similarity on a smaller subset of items we tackle a problem related to users, who rated many items. At first glance, such users should be easier to serve, and will be provided with better recommendations, because they have a large rating history. However, when looking for neighbors it

¹Thanks to J. Konstan for suggesting this study in a personal communication.

is likely that there will be many users highly correlated to this target user but only on a small number of overlapping items. At the end, without item selection, we could select neighbors who are highly correlated but on a small number of items. Using item selection we compute overlap only on few but **highly selected** items, making more likely that other users with larger profiles become neighbors of the target user. This may decrease the average correlation with the neighbor users but it computes correlations with higher reliability.

Moreover, since the selection of items is made by analyzing both the target user and the neighbor profile, the neighbors found are more diverse, as they can be correlated on different subsets of items. This is very important and is explaining the increase in recall as the increased diversity of the neighbors increases the chance that a neighbor rated the target item.

6.3 Discussion and Conclusions

In this chapter we have analyzed a wide range of item weighting and two item selection techniques suitable for CF. Each item weighting technique analyzes the data to get additional statistics about the relationships between the items. This information is then used (via item weighting or item selection) to fine-tune the user-to-user similarity and to improve the accuracy of the CF recommendation technique. We have observed that the newly-introduced SVD-PCC weighting schema — designed to work for sparse data — performs better than the baseline CF method in the two datasets that were considered, and for all the accuracy measures that we used (MAE, F, Coverage). All the other methods do not have such a stable behavior and they outperform the baseline CF only for some of the error measures, and sometimes they perform even worse. We also experimentally evaluated three different weight incorporation techniques and explained their behavior. In the data sets that we used $WPCC_{no-norm}$ showed a better performance than the other techniques.

In this chapter we also introduced locally adaptive item selection method for an improved neighbor selection in CF. We evaluated the proposed method along with a range of error measures. In summary, BIPO item selection approach, using a small number of items, can achieve significant improvements for all the considered error measures. This result is important because it shows that CF user profiles contain a lot of redundancy and, even if the data set is sparse, the information is not uniformly distributed among users.

We showed that CF performances can be improved with a careful selection of the item ratings, i.e., acquiring ratings for certain items can greatly improve the performance of the recommendation algorithm. In fact, since BIPO selects items according to the target item, the outcome can be a different neighborhood for the same target user depending on the prediction that must be computed.

Hence BIPO is in fact an item selection and user selection method. The idea is that when we must make a prediction for a particular type of item, not all the neighbors computed by the classical CF may be relevant. If a neighbor user is highly correlated to the target, but on items not highly correlated with the target item, then this user should not be considered.

Part III

Implementing CACF

Building Context-Aware Collaborative Filtering Systems

This chapter describes and analyzes a prototype CACF - a real time context-aware Collaborative Filtering recommender system for mobile users. We built and investigated this system during my Ph.D. studies. Here we propose a methodology for supporting the development cycle of a CACF system. It also provides an example of application that could be used together with CACF rating prediction techniques presented in the Part II of the thesis.

7.1 Challenges of Building CACF System

To adapt the recommendations to the user's current contextual situation requires an understanding of the relationship between user preferences and contextual conditions. In many recommender systems, especially those based on Collaborative Filtering, the user preferences are expressed via item ratings. Therefore, to model the relationship between ratings from context, it has been proposed that explicit user item ratings should be captured under several different contextual conditions [Adomavicius et al., 2005, Adomavicius and Tuzhilin, 2010]. For instance, the user must rate the suggestion to visit a Museum when she has children with her (contextual factor: *composition of the group of visitors*, contextual condition: *visit with children*), when she does not have children with her, when it is raining / not raining etc.. Such data is expensive to obtain in naturalistic experiments (where the user is given suggestions as they go about their lives as normal) because it requires the same recommendations to be rated in multiple contextual situations. The user must provide item ratings after those items have been *experienced* in several different contextual conditions. Therefore, it is important to minimize the risk to set up an expensive process for acquiring such ratings and then discover a

posteriori that the selected contextual conditions were in actual fact irrelevant or not important to investigate.

7.1.1 Context-Aware Recommendations are Difficult to Compute

Hence, a major issue for the successful design of CARSs is the discovery of the contextual factors that are worth considering when generating recommendations. This is not an easy problem to solve. It requires formulating informed conjectures about the influence of certain factors before collecting data in naturalistic environments. It is a kind of active learning problem, where the relevance of the data to acquire must be estimated to minimize the cost of the real data acquisition phase [Rubens et al., 2010]. This estimation of the relevance of contextual factors is the first problem for building CARSs.

After a meaningful set of contextual conditions is identified, a predictive model must be built that can predict how the evaluation of an item changes as a function of the contextual factors [Amatrian et al., 2010]. This model is then used to select items given a target context. This step again requires the collection and utilization of explicit ratings for items under several distinct contextual conditions. In Collaborative Filtering (CF) this means, as we mentioned above, the collection of ratings in context. The acquisition of a representative sample of in-context item ratings is the second problem for CARSs.

Finally, after the model is built a complete context-aware recommender system can be generated. By this we mean a complete human computer interaction layer can be designed and implemented on top of the core predictive model. The user should be able to query the system for recommendations, specifying preferences and contextual conditions, and should receive useful suggestions that will be actually acted on [McNee et al., 2006]. In a travel guide application, for instance, the user may request recommendations adapted to the precise day of the trip, a specific starting location, and the fact that his family want to travel by bike. Computing good recommendations efficiently on the basis of a given predictive model is the third problem for CARSs.

Moreover, the system must adapt the recommendations to this request and provide an effective visualization of the recommendations, including useful item descriptions, and explanations for the recommendations [Herlocker et al., 2000, Mcsherry, 2005, Tintarev and Masthoff, 2010]. Visualization and related issues for the user interface therefore are the fourth problem for CARSs.

7.1.2 Towards a Methodology for Developing CARS

In this chapter, we propose systematic solutions for the four problems introduced above. The main contribution therefore is a methodology for supporting the

development cycle for CARS. This methodology comprises four steps: determining which contexts are interesting to study; acquiring user ratings in specific contexts of interest; predicting ratings given a specific context; context-aware recommendation visualisation and updating. Each of these steps is supported by a specific system or algorithm, which we briefly illustrate in this chapter.

First, in order to quantitatively estimate the dependency of the user preferences from an initial candidate set of contextual factors we developed a web tool for acquiring context relevance subjective judgements. To achieve this, the user is requested to evaluate if a particular contextual condition, e.g., “it is raining today”, may have a positive or negative influence on the user’s rating of a particular item. We analysed the resulting data statistically to establish which contextual factors are most likely to influence the user decisions for different types of item.

Second, we developed a further web application that generates example contexts and recommendations for these contexts, which the participants were asked to rate. The generation process was informed by the results of the previous step. The more relevant a contextual factor is according to the results of the first data collection, the more often contextual conditions specifying this factor are generated for rating requests in the second system. For instance, because the first results show that the distance to a castle is a very relevant contextual factor for that POI type, in the second application the users are very likely to be requested to rate a recommendation to visit *Mareccio Castle* assuming that they are either close to or far from it. The ratings collected in this evaluation, as with the previous step, were provided by the users not while experiencing the items in these contextual conditions but by imagining the situation and providing a judgement. The advantage of this approach is that ratings can be recorded for multiple relevant contextual conditions, without any constraint related to what the user has actually experienced in the past. This is not possible using the approach typically used to evaluate Context Aware Recommender Systems based on Collaborative Filtering.

Third, a predictive model is built, the goal of which is to predict the user’s ratings for items under other contextual situations where these ratings are not known. Recommendations, as usual, are then selected among the items with the largest predicted rating for the context situation of the target user. Here a contextual situation is a combination of several individual contextual conditions, e.g., the user is asking a recommendation in a rainy day (condition one) when he is traveling with children (condition two). This model computes the influence of each contextual condition on the prediction, and therefore can also be used to isolate the contextual conditions that have a larger effect either in increasing or decreasing the predicted ratings for items.

Finally, the predictive model is exploited in a mobile application, called ReRex – a travel planning application aimed at recommending points of interests

(POIs) to mobile users. It offers two functionalities. Firstly, context-dependent and personalized recommendations of touristic POI. Secondly, assistance in the preparation of a complete itinerary and the modification of the itinerary according to circumstances and eventualities that occur during the itinerary. In particular, the user can request recommendations for a particular context. The application presents the recommendations generated by the predictive model and justifies the recommendations with a direct and simple explanation of the main reason why an item is recommended for that particular contextual situation. So, referring to the example mentioned previously, the system may justify the recommendation to a particular POI (e.g., Oetzi Museum) because the user is traveling with children. Further, the mobile application can asynchronously notify the user that a contextual condition (e.g., weather) is likely to change and therefore revises the recommendations for the user.

ReRex mobile application was tested in a live user experiment to validate the full process defined by the methodology proposed above. The methodology described in this chapter is pretty general and can be applied to recommender systems in various domains. A more detailed description of the Web interface and the methods to acquire the relevance of context can be found in [Baltrunas et al., 2011b]. In fact, we are currently also testing the approach on an in-car context-aware music recommendation scenario.

7.1.3 ReRex: The New Methodology in Practice

In conclusion, in this chapter we focus on the discussion of our methodology for building context-aware recommender systems (CARS) and the presentation of ReRex, our prototype for putting the methodology into practice.

In Section 7.3, we explain our approach for acquiring user data needed for estimating relevant contextual factors. We claim that the contextual factors that are worth considering in a CARS must be assessed before the in-context evaluations are collected. In fact we discover that the relevance of a contextual factor can be user and item specific. We show how the relevance can be acquired by asking users to evaluate if a contextual condition for that factor has a positive or negative impact on his decision to select the item.

In Section 7.4 we present the methodology that we have applied to acquire in-context item ratings, i.e., assuming that a certain relevant contextual condition holds. We illustrate how useful in-context evaluations can be acquired by asking to users to imagine that a contextual condition holds and then to rate an item. Even if these ratings may differ from those that would be collected after the user has really experienced the item in the context [Ono et al., 2009, Asoh et al., 2010], they nevertheless help to model the expected appropriateness of an item [Ricci et al., 2010].

In Section 7.5 our predictive model for context-aware recommendations,

which extends Matrix Factorization, is presented and its accuracy is evaluated off-line with the acquired rating data. We show that using the proposed model both personalization and context-awareness can substantially improve the model accuracy.

In Section 7.6 we introduce our context-aware mobile recommender system prototype ReRex. ReRex offers places of interest recommendations, but also identifies the contextual condition with the greatest impact (increase) on the predicted rating of a recommended item. This condition is used as an argument to explain what makes the item better suited in the current user situation.

ReRex is evaluated in Section 7.7. It is shown that ReRex received a better evaluation compared with a similar, but not context-aware system. We observe that in this evaluation recommendations were generated for the target users without knowing any of their ratings for items. In fact, ReRex used a context-aware but not personalized recommendation model (a variant that is supported by our rating prediction technology). Hence we show that effective recommendations can be computed knowing only the user's current contextual situation. The chapter ends with the conclusions we have drawn from this study, and lists some open issues that call for some future work.

7.2 Context Dependent Data Acquisition

Before going into the details of the proposed methodology we first explain the general problem and possible approaches of context dependent data acquisition.

Research on CARS is taking for granted that context matters, i.e., that recommendations should be adapted to the specific situation in which the recommended item will be consumed. So, for instance a recommendation for a holiday destination should take into account if the travel will occur in summer or winter. But, several attempts to incorporate contextual information in the generation of the recommendations have not shown any significant improvement compared with simpler non context-aware approaches.

Clearly, there could be domains where context is irrelevant, and there is no benefit of using any additional information beyond the user ratings. However, even in those domains where the user context has a clear influence on the user behavior, e.g. tourism, it is not trivial to build a context-aware Recommender System. The first problem relates to the context-aware data availability. We note that not all of the contextual factors can be detected automatically; for instance the user companion or travel goal should be obtained by querying the user. So, in general this means that some contextual conditions should be manually specified by a user, and the system should compensate for this additional work with more relevant recommendations [Herlocker and Konstan, 2001]. This yields a circular problem where the system has not enough contextual information to generate

better recommendations and user is not providing such information as it is not worth for the additional work. One obvious way to break this deadlock is to bootstrap a context-aware system by collecting initial data [Ono et al., 2009].

In the classical reduction based context aware collaborative filtering approach [Adomavicius et al., 2005] one should sample the rating space, i.e., asking to a population of users to rate items only in the contextual conditions that users *have experienced* the item in. A second approach consists of considering ratings assigned by users in hypothetical contextual conditions, i.e., evaluations that the user assign to the items imagining to be in a target context [Ono et al., 2009]. For instance, asking the user to imagine that it is raining, and then asking him to evaluate his intention to visit a particular POI. In our work we used this approach as it is clearly less time consuming. In addition, a single user could give several ratings for an item in different contextual conditions even he had experienced the item only once. This is a big advantage, as the rating matrix becomes more dense and the item consumption by the same user can be compared in different contexts. Such comparison gives additional information that could be used to learn the context relevance for the item and the user. We call both approaches **rating in context** data acquisition as the rating that is collected can be interpreted in a given context. For example, our system asks a user to imagine that he is in Bolzano and is making a touristic plan for today. It asks the user to estimate how likely is that he will visit a POI (such as Messner Mountain Museum) in a 5 star rating scale. Next, the user has to imagine that it is raining and the system asks again to indicate how likely is that he will visit the same POI. This gives us explicit evaluations that could be used in a rating prediction algorithm to estimate the ratings in any contextual condition.

7.2.1 Overview of the Implemented Data Acquisition Process.

Having limited resources, i.e., a restricted number of users to survey and a limited amount of time, it is clearly not possible to collect ratings for all the items in all the possible contextual conditions. For instance, in the prototype system that we will describe now, there are 14 contextual dimensions, and each one has several possible values. Therefore, the number of possible contextual combinations exceeds 34M cases. It is clearly not feasible to collect ratings in all the possible contextual conditions. Thus, in order to build a successful CARS a simplified and more systematic approach should be taken.

Moreover, one should minimize the risk to set up a process for acquiring such ratings and then discover a posteriori that the selected contextual conditions were actually irrelevant. Namely, one can discover that the user's ratings are not influenced by the selected contextual conditions, or that this information does not help in improving the recommender system effectiveness [Jeong et al., 2009].

Hence, a major initial issue for the correct design of CARSs is the assessment

of the contextual factors that are worth considering when generating recommendations. We have elaborated a system design methodology where we assume that users can be requested to judge if a single contextual condition is relevant for their decision making task, and how they would evaluate an item assuming that certain contextual conditions hold. Therefore, we propose a two steps rating in context acquisition procedure:

- Firstly, in order to estimate the dependency of the user preferences from an initial candidate set of contextual dimensions we developed a tool for acquiring context relevance subjective judgements. In these judgements the user is requested to evaluate if a particular contextual condition, e.g., “today is raining”, may have a positive or negative influence in his propensity to select a particular type of items. Furthermore, we identified and applied a data analysis method for selecting the contextual dimensions that are more likely to influence the user decisions for different items’ types. This step and obtained results are described in detail in Section 7.3
- Secondly, we developed a user interface that generates rating requests to the users. The generation method is informed by the context relevance results obtained at the previous step. Indeed, the larger the influence of a contextual situation for an item type is, the more likely the user is requested to provide a subjective rating for that item type in that contextual situation. In other words, the system collects in context ratings but sampling earlier the contextual situations that are more likely to have a larger impact (either positive or negative) on the in-context evaluation. We observe that these evaluations are, as for the previous step, provided by the users not when they are experiencing the items in these contextual conditions but when they are imagining to be in that situation. Hence, these requests can be made for all the possible relevant contextual conditions, without any constraint related to what the user actually experienced in the past (standard approach in other CARS systems). This step and results are described in detail in Section 7.4

7.3 Acquiring Context Relevance

The first step of the methodology described in the Introduction pertains to the computation of a quantitative measure of the influence of some potentially relevant contextual factors on the user item selection decisions. To attain this objective we collected data how users *change* their inclination to visit a POI while they imagine that a certain contextual condition, i.e., a specific value for a contextual factor, holds. For that purpose, we designed an online Web application.

Context Factor	Values	Context Factor	Values
budget	budget traveler high spender price for quality	crowdedness	not crowded crowded empty
time of the day	morning time afternoon night time	travel goal	health care cultural experience scenic/landscape education hedonistic/fun social event religion
day of the week	weekend working day		activity/sport visiting friends business
distance to POI	near by far away	season	spring summer autumn winter
knowledge about area	new to city citizen of the city returning visitor	transport	public transport no means of transport bicycle car
companion	with girl-/boy-friend with family with children alone with friends	temperature	warm cold hot
weather	snowing clear sky sunny rainy cloudy	time available	half day more than a day one day
mood	happy active sad		

Table 7.1: Overview of the context factors used in the web survey

First, a large set of contextual factors and conditions (values for the factors), as found in the literature on consumer behavior in tourism was selected [Swarbrooke and Horner, 2007]. The selected contextual factors and conditions are listed in Table 7.1. We observe that an alternative approach for identifying potentially relevant contextual conditions could be a qualitative study, such as a diary study. Then, a relatively small list of types of POIs in Bolzano (Italy) and other nearby towns were identified. POIs were aggregated into categories in order to avoid sparseness of the collected data; that is, we assume that the influence of a contextual factor is uniform for all the POIs in a category. We defined ten categories: castle; nature wonder; cycling and mountain biking; theater event; folk festival, arts and crafts event; church or monastery; museum; spa and pampering; music event; walking path.


Then, we have developed a simple Web application (see Figure 7.1) for acquiring the relevance of the selected contextual factors for the POIs categories. In this Web application, the users are requested to evaluate the influence of the selected contextual conditions on their decisions to visit the POIs belonging to a randomly selected item category. As an example of the questions posed to the user consider the situation depicted in Figure 7.1. Here we first request to the user to imagine a typical visit situation: “Imagine that you are in Bolzano making a plan for today. You are considering to relax in a spa”. Then we asked to select the influence of three randomly selected contextual conditions on that visit. The influence can take one among three possible values: positive, negative or neutral. As an example of a contextual condition: “Imagine that it is a cold day”. So the user was requested to evaluate if this condition increases, decreases, or has no influence on his decision. Every user was requested to interact with at least five of these pages (as in Figure 7.1).

33 participants (mostly from our computer science faculty) took part in this web survey. Overall, they gave 1524 responses to one of the questions shown in Figure 7.1. For the specification of the context, the factors presented in Table 7.1 were applied in a randomized way: for each question a POI category was drawn at random along with a value for a context factor. This sampling has been implemented such that a uniform distribution over the possible categories and the possible contextual conditions is achieved. A different sampling could also be applied if a prior distribution is known. In this way one can try to obtain more information on the conditions that are more likely to occur or that are more likely to influence the user’s decision.



7.3.1 Analysis of Context Relevance

The web survey delivered samples for the probability distributions $P(I, C_i, T)$; where C_1, \dots, C_N are the context factors that (may or may not) influence the

(user:ciccio)logout



Imagine that you are in Bolzano and you are making a plan for today. You are considering **to relax in a spa**. Please mark the conditions that would positively or negatively influence the decision to do that, or would have no effect.

		No effect	
Imagine that you are on a wellness trip:	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>
Imagine that it is a cold day:	<input type="radio"/>	<input checked="" type="radio"/>	<input type="radio"/>
Imagine that it is raining:	<input type="radio"/>	<input type="radio"/>	<input checked="" type="radio"/>

[next...](#)

Situation 1 of 5

Figure 7.1: Web based survey tool to acquire context relevance.

user decisions, T is a POI category, and I (Influence) is the response variable, taking one of the three values: positive, negative, or neutral. The probabilities $P(I|C_i, T)$ model the influence of the context factors on the user's decision. The knowledge of $P(I|C_i, T)$, as we will discuss in the next section, can drive the acquisition of context-dependent ratings for the context factors that have a large influence on the user evaluation for the items in a given category T . Hence, it is interesting to measure, given a category T , the impact of context C_i on I , and then which C_i best explains I .

The spread of a categorical variable $X = \{x_1, \dots, x_n\}$ can be measured with its entropy [Lloyd, 1999]. If $P(X = x_i) = \pi_i$, the entropy of X is defined as follows:

$$H(X) = - \sum_{1 \leq i \leq n} \pi_i \cdot \log \pi_i$$

This measure of the spread can be used to estimate the association of two variables, e.g., I and C_i , i.e., how much the *Inclination of the user to visit an item*, is influenced by the context of the current situation, e.g. the *Current weather condition*. Informally, this influence is strong if the knowledge about the weather reduces the spread of I , and it is weak if the spread of I remains unchanged even if one knows the weather. Therefore, the difference between the spread of I and the expected spread of $(I|C_i)$ is a measure for the association of the two variables. As the spread of $(I|C_i)$ should not be larger than that of I alone we can normalize the difference to the interval $[0, 1]$ by:

$$U = \frac{H(I|T) - H(I|C_i, T)}{H(I|T)}$$

U is 1 if the spread of $(I|C_i, T)$ is zero. This occurs if for each value of C_i the value I is certain. Conversely, U is zero, if C_i does not have any influence on I , in which case the spread of $(I|C_i, T)$ is not different from that of $(I|T)$.



Hence, in order to understand which context factors help most to decrease the uncertainty about I , we have computed U for all factors and POI categories. Ordering the factors in descending value of U , one gets the results reported in the Appendix B of the thesis. That table indicates that there are some factors that indeed are relevant for all the categories, among them *distance to POI*, *time available*, *crowdedness*, and *knowledge of the surroundings*. Others often appear to be less relevant: *transport*, *travel goal*, *day of the week*. Finally some factors appear to have a different relevance depending on the category. We want to note that most of the findings factors are expected. However, some examples were less obvious: visiting museum in a hot day decreased the likelihood to visit it and being lazy increased the likelihood to go for a walk. We want to stress that even knowing the general tendencies of a contextual factor on a POI it is hard to quantify how much they should affect the ranking.

7.4 Acquisition of Ratings in Context

In the second phase of our study, as we wanted to measure how a POI's rating is actually modified by the contextual conditions. For that purpose, we considered a set of POI in Bolzano, namely 20 POIs, and asked to users to rate them under different contextual conditions. The same POIs have been also used in the mobile recommender system, ReRex, that is described in Section 7.6. In fact, in order to be able to measure the *variation* of the rating for a POI when a contextual condition holds or not, in each interview we asked the user to rate a POI in two situations: without considering any contextual condition and assuming a particular contextual condition to be true. Figure 7.2 shows a screenshot of the

Rating in Context

Castel Flavon - Haselburg

Category: castle

Introduction: Castel Flavon Haselburg nestles on a wooded hill slightly above Haslach, a quarter of the city of Bolzano. Built in late 12th century, still today it boasts some valuable frescoes.

Description: Castel Flavon Haselburg nestles on a wooded hill slightly above Haslach, a quarter of the city of Bolzano. Built in late 12th century, still today it boasts some valuable frescoes. It was recently renovated and the restaurant is open again. Address: Via Castel Flavon 48 Phone:0471 402130 Email: info@haselburg.it www: www.haselburg.it Opening hours: Tuesday-Saturday 11am-12pm, Sunday 11am-5pm, Monday closed.

Imagine you are in Bolzano and you are making plan for today

How likely is that you will visit Castel Flavon - Haselburg ★ ★ ★ ☆ ☆

We want to know which circumstances influence your decision

Imagine that you are sad. How likely is that you will visit Castel Flavon - Haselburg: ★ ★ ☆ ☆ ☆

Imagine that you feel comfortable and happy. How likely is that you will visit Castel Flavon - Haselburg: ★ ★ ★ ★ ☆

Imagine that you can only use public transport. How likely is that you will visit Castel Flavon - Haselburg: ★ ★ ★ ☆ ☆

Next

Figure 7.2: Web based survey tool to acquire ratings in context.

Web tool that we implemented for this task. In an interview a single POI is considered, and four ratings are requested: first, in general, how likely is the user to visit the POI, and then the same evaluation but assuming that three alternative contextual conditions are given. In these interviews each contextual factor is drawn from the full set of factors proportionally to its relevance U , as computed in the previous phase. We collected 371 of such interviews from 24 different users, that live in Bolzano city, hence a total of 1484 ratings were acquired for 20 POIs (one fourth of them without a context specification and three fourth with a contextual condition specified). We observe that these ratings have been used to build the rating prediction model (described in Section 7.5) and to generate the recommendations in the ReRex mobile application (described in Section 7.6).

It is worth noting that, by construction, we collected more ratings for in-context evaluations related to the most influential contextual factors. The

rationale is that these factors are predicted, based on the previous analysis (Section 7.3), to have a larger impact on the ratings, and therefore should bring more useful information. But, this has also another effect on the predictive model – to be described in Section 7.5: the model accuracy will be higher for predictions under these contextual conditions.

In the table in Appendix B, we present the analysis of the data collected in this phase for a set of well known points of interest in Bolzano. In this table, *Mean context yes* (MCY) is the average rating for all items in the specified category assuming that the given contextual condition holds. *Mean context no* (MCN) is the average rating for the same items without any contextual condition to hold. The given p -value is that of a t -test¹ comparing MCN to MCY. Please note that in this table we list only the contextual conditions with a significant difference of the two means MCN and MCY.

It is notable that in the majority of the cases, context has a negative influence on the ratings of the users. In several cases in which the rating in context is higher the p -values are not significant. To find a plausible interpretation for this observation, we recall from psychology that people are more sensitive to negative than to positive influence when making decision under uncertainty [Tversky and Kahneman, 1991]. Hence, the user may remember more vividly the negative impact of context rather than the positive one.

Furthermore, these results may be a consequence of the artificial setup of the experiment: doing the survey in front of a desktop PC may lead to an overvaluation of exceptional situations as they occur in some interviews. Another explanation is that most POI have a high rating without context. As a consequence, in such a situation, for the user there is no way to rate a POI higher even if a context factor would support the user’s positive attitude towards this POI.

A detailed look at the data reveals the fact that significant negative ratings can be found very often if the contextual condition under consideration describes an ‘extreme’ situation for visiting the POI or doing the recommended activity. For example: crowded and empty are both valued negatively while the rating for ‘not so crowded’ is above the rating without context (however without significance).

Furthermore, ratings are significantly negative if the user probably supposed the factor would lower the value of the POI or decrease the user’s comfort. E.g. ‘far away’ leads to very low ratings of POI – as the factors ‘half a day’, or ‘car’ (maybe users assume a car would not be helpful to visit a POI in the city centre).

Context values may also be in conflict with user preferences. For instance, The contextual factor *travel goal* can take the two values *scenic/landscape* and *sports/activity*. In these contextual conditions many POIs received low ratings

¹For the ratings we assume a normal distribution; therefore the t -test is appropriate to detect significant deviations of MCY from MCN.

because located in the city centre and in general they are not suited for people who would prefer to do sport activities like trekking or climbing (very popular among the users that tried the system).

To summarize these observations: the table illustrates that for many contextual conditions statistically significant differences can be detected in the ratings compared to the ratings of the same points of interest without that conditions. We conclude that context-dependency must be taken into account and is likely to improve the accuracy of the recommendations compared to a non-context-aware system. The off-line study described in the next section and the live user evaluation of our mobile application ReRex (see Section 7.6) provide empirical evidence for this claim.

7.5 Context-Aware Recommendation Algorithm

An important component of the ReRex mobile context-aware recommender system, which will be illustrated in this section, is its rating prediction and recommendation algorithm. The rating prediction component computes a rating prediction for all the items, while assuming that the current user context holds. Note that the prediction algorithm proposed here could be in principal replaced by the algorithms proposed in Chapter 3 or Chapter 4. However, as we will see, it is specifically design for our gathered data, moreover, has effective explanation heuristic.

The current context is partially specified by the user, using the system GUI, and partially acquired automatically by the system (as explained in the next section). Then the items with the highest predicted ratings are recommended. In this section, we will present an extension of the Matrix Factorization (MF) rating prediction technique that incorporates contextual information to adapt the recommendation to the user target context. We have designed and tuned our MF extension considering some additional requirements that we deem important for our application. In particular, the main features of our algorithm are:

- **Trainable with different types of data.**

As we saw before, the output of the data collection procedure is a set of subjective user ratings for items under specific contextual conditions – or without any consideration of the context (default context). Each user rating is actually tagged with one contextual condition only. To give an example, the visit to Castle Mareccio is rated 4 when this POI is not far away from the current user position. But, when a user is actually using the mobile application (as described in the next section) she may also provide a rating. In this case the rating is recorded with all the other contextual conditions (e.g., weather) acquired by the system for that situation (and considered as

relevant by the user). Therefore, this user rating may be associated with zero, one, or *more* contextual conditions. The rating prediction algorithm that we have designed is able to train the rating prediction model using both kind of data indifferently.

- **Update of the predictions when the context changes.**

The contextual conditions considered in our application can rapidly change. Moreover, our application allows the user to enable or disable some of the contextual factors (see Section 7.6). This means that it is up to the user to decide if the system must take into account a contextual factor or not. In fact, we believe that this is an important feature of context management, since as we have shown before, it could be rather difficult for a completely autonomous system to decide about that. Consequently, the predicted recommendations should be adapted, to any change of the contextual situation. This requires a method that can compute context-aware recommendation predictions in a short time.

- **Explanations for the recommendations.**

Explanations can have several positive effects on the system usability – among them enhancing the system transparency, its persuasiveness, and the user trust on the system [Tintarev and Masthoff, 2010]. We believe that referring to the contextual conditions under which the recommendations are generated can have a strong effect on the user acceptance of the recommendation. For example, explaining that a visit to a museum is explicitly recommended because today is a cold day, can add a good and possibly crucial argument for the acceptance of the recommendation. Therefore, we included in our technique a component that isolates a single contextual condition, among those relevant in the current situation, and motivates the recommendation by referring to that contextual condition as a reason for recommending the item.

7.5.1 The Predictive Model

The above listed features have shaped our algorithmic solution. As we need to compute recommendations for rapidly changing contextual conditions, we selected a model-based rating prediction approach. The predictive model is therefore trained off-line; once every hour or when a meaningful number of new ratings are collected. The trained model is then used to generate recommendations. The important feature of this approach is that it can generate rating predictions in a constant time². Before describing how the other system features have been

²In fact, as we will see later on, it is constant with respect to the number of input data and linear in the number of contextual conditions.

implemented we must first describe the details of the predictive model.

In [Koren, 2008] the author presents a Matrix Factorization approach to CF that uses “baseline” parameters for each user and item. Baselines are additional model parameters that are introduced for each user and item. They indicate the general deviation of the ratings of a user or an item from the global average. So for instance, the user baseline, which is learned for a user that tends to rate higher than the average of users’ population, will be a positive number. Additional parameters like these slightly increase the model complexity. However, they can also improve its accuracy, as we will show later. Baseline parameters also serve for taking the impact of context into account. This has been already shown by [Koren, 2009], where the author has introduced several baseline parameters to model the time dependency of the ratings.

We have extended and adapted that approach, and we have incorporated more contextual dimensions into the MF model. We propose to introduce one model parameter for each contextual factor and item pair. This allows to learn how the rating deviates from the classical personalized prediction as effect of the context. This deviation is the *baseline* for that factor and item combination. Broadly speaking, the system computes a rating prediction for a user-item pair and then adapts that to the current context using the learned context-dependent baselines. Suppose that the user u rates the item i as r_{ui} . Now the user is asked to rate the same item i in a given contextual condition c , producing the rating r_{uic} . The difference of the two ratings, $r_{ui} - r_{uic}$, indicates how much c influences the rating and this information can be exploited in the model learning stage. Clearly, this is an oversimplified example, as there is typically noise in the data. Moreover, the model must simultaneously learn other parameters for correctly modeling the dependency of the rating on u , i and c . In our data set of context-aware ratings, a rating $r_{uic_1 \dots c_k}$ indicates the evaluation of the user u of the item i made in the context c_1, \dots, c_k , where $c_j = 0, 1, \dots, z_j$, and $c_j = 0$ means that the j -th contextual condition is unknown, while the other index values refer to possible values for the j -th contextual factor. The tuples (u, i, c_1, \dots, c_k) , for which rating $r_{uic_1 \dots c_k}$ is known, are stored in the data set $R = \{(u, i, c_1, \dots, c_k) | r_{uic_1 \dots c_k} \text{ is known}\}$. Note, that in our collected data set, only one contextual condition is known and all others are unknown, hence in R there are ratings for which only one among the indices c_1, \dots, c_k is different from 0.

We recall that MF aims at factorizing the ratings matrix into two $m \times d$ and $n \times d$ dimensional matrices V and Q respectively. A user is then represented with a column vector $v_u \in V$ and an item i with a column vector $q_i \in Q$. One can balance the capacity and generalization capability of the predictive model by increasing or decreasing the dimensionality d of V and Q . We propose to compute a personalized context-dependent rating estimation using the following

model.

$$\hat{r}_{uic_1 \dots c_k} = v_u q_i^\top + \bar{i} + b_u + \sum_{j=1}^k b_{ic_j} \quad (7.1)$$

where v_u and q_i are d dimensional real valued vectors representing the user u and the item i . \bar{i} is the average of the item i ratings in the data set R , b_u is the baseline parameter for user u and b_{ic_j} is the baseline of the contextual condition c_j and item i . If a contextual factor is unknown, i.e., $c_j = 0$, then the corresponding baseline b_{ic_j} is set to 0. In this way, one can learn the influence only of the known contextual conditions.

This model needs further explanations. Clearly, when designing any model one must take into account the amount of available training data and guess how complex the relationships in the underlying data are. The model could be extended also to take into account dependencies between contextual factors. At the moment, context's influence on the item is modeled with a parameter b_{ic_j} . We could add additional parameters $b_{ic_j c_k}$ that would model the dependencies between contexts c_j and c_k . More complex models could better fit the data. However, if there are not enough training data, this can have a negative effect. Since we do not have a large amount of bootstrapping data, in comparison to other data sets such as Netflix³ or Movielens⁴, we opted for this simple linear model that is shown in Equation 7.1. In that model the contextual information is exploited by using a set of model parameters, b_{ic_j} . In the learning phase, which is illustrated later on, the influence of the contextual factor on the ratings is fully described by these parameters.

Note, that this model assumes the independence of the contextual factors. This could appear as a limitation, as one can easily find examples that do not satisfy this assumption. However, this assumption was implicitly made when we collected the training data as we asked the users to provide their ratings assuming a single contextual condition per time. Hence the model bias fits the data acquisition bias. Nevertheless, the model can capture the influence of multiple contextual factors in the prediction stage but can still be learned with our particular training data. This is done by summing up over all the baselines of the known contextual factors. In the future we plan to test this model also on larger and maybe more complex training data.

The flexibility of this approach is also shown by the observation that the proposed model could be extended to incorporate more information, if available. E.g., adding a parameter that depends only on the context, but not on item, would allow the system to learn more general rules about context, such as, on a sunny day all the items are better rated. Adding such kind of dependency can make the model even more accurate without increasing its complexity, but will

³www.netflixprize.com

⁴www.movielens.org

have *no effect* on the items' ranking, since the ratings of all the rating predictions will be altered by the same quantity.

7.5.2 Model Training

In order to generate rating predictions, the model parameters should be learned using the training data. We define the learning procedure as an optimization problem:

$$\min_{v_*, q_*, b_*} \sum_{r \in R} \left[\left(r_{uic_1 \dots c_k} - v_u q_i^\top - \bar{r} - b_u - \sum_{j=1}^k b_{ic_j} \right)^2 + \lambda (b_u^2 + \|v_u\|^2 + \|q_i\|^2 + \sum_{j=1}^k b_{ic_j}^2) \right]$$

where $r = (u, i, c_1, \dots, c_k)$. For better generalization performance, a regularization term is added, as it is usual in this type of models. Regularization is controlled by the λ meta parameter. As λ grows the model becomes more “rigid”, and fits less the variability in the training data. We have used stochastic gradient descent for solving this problem. This has been proved to be an efficient approach [Koren, 2008]. We update all the parameters of the model as follows:

- $b_u \leftarrow b_u + \gamma_{b_u} (err - \lambda b_u)$
- $b_{ic_j} \leftarrow b_{ic_j} + \gamma_{b_{ic_j}} (err - \lambda b_{ic_j}), \forall c_j \neq 0, j = 1, \dots, k$
- $v_u \leftarrow v_u + \gamma_{v_u} (err \cdot q_i - \lambda v_u)$
- $q_i \leftarrow q_i + \gamma_{q_i} (err \cdot v_u - \lambda q_i)$

This procedure updates one after another all parameters that are moving in the opposite direction of the gradient, while all the other variables are kept unchanged. Actually, to be precise, the first entry of the two vectors v_u and q_i is learned until these vectors do not further improve the model accuracy. Subsequently, the second entry is adapted in the same way. This process is repeated until all the entries are considered. The learning rate depends on the meta parameter γ that was set in our experiments to 0.001. For each rating in the training set we computed the error, i.e., the difference between the actual and the predicted rating $err = r_{uic_1 \dots c_k} - \hat{r}_{uic_1 \dots c_k}$. The larger the error, the bigger is the adaptation of the parameters, and the sign of this error indicates in which direction the adaptation must be performed.

Note that some of the parameters in the model are updated more frequently than others. This is due to the fact, that some users have more ratings, or some

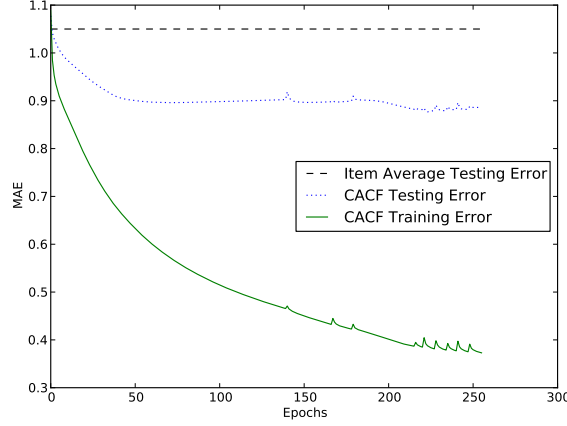


Figure 7.3: Training and Test Errors

contextual factors appear more often. This could make the model unbalanced as more frequent data will be overrepresented in it. To solve the problem we adapted the learning rate γ , computing γ_* (e.g., γ_{b_u}). γ_* is obtained dividing γ by the logarithm of the number of updates of the corresponding parameter. For example, imagine that a user u has 20 ratings in his profile. In this case learning rate for variable b_u is $\frac{\gamma}{\log(20)}$. We observe that still will be influenced more by the most popular contextual factors in the rating data set, as we normalize with the log of the frequency and not with the frequency. This is important as these are also the most influential factors (as we discussed earlier).

In this learning procedure the parameters are updated while cycling through the data. In Figure 7.3 the model error while executing the learning procedure is shown. On the x axis we show the number of runs through the data that we call epochs. In an epoch all the training data are used once. Here the training data comprise a random selection of 80% of the total and the remaining part is used for testing. For a baseline reference we have also plotted the error of a simple model that predicts the user rating for an item as the average of the ratings for the item provided by the other users. This is a non-personalized, non-contextualized approach. It can be clearly seen that the more train epochs are executed, the better is the performance on the training set. However, as usual, we see a different behavior on the test set. At the beginning, the error decreases, but at some point it starts to increase again. This overfitting could be avoided by increasing the value of λ or learning less parameters for the user and item representation. Note that sometimes one can see a small increase in

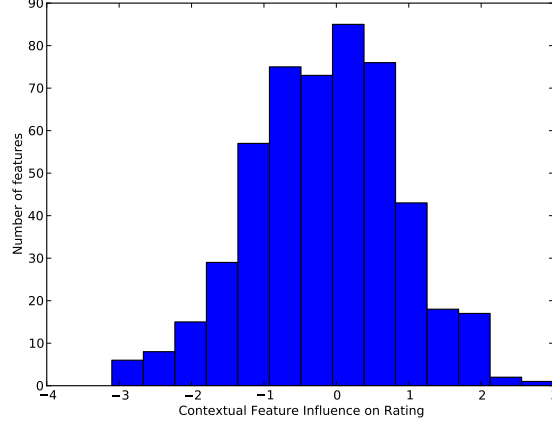


Figure 7.4: Learned Contextual Influence.

the error. This is the point where the algorithm starts to learn new parameters for a user and item vectors (v_u and q_i). As these parameters are initialized with random values, this causes a temporal increase of the error. We observe that in our experiments we used $d = 10$ factors for the item and user representation v_u and q_i , since using more factors did not improve the quality of the model.

We have also analyzed the performance of some variations of the proposed model. We observed how much each additional set of parameters influences the model accuracy. The model was altered by removing some of the parameters. For example, the model for a non-personalized, but contextualized rating prediction is defined as follows:

$$\hat{r}_{uic_1 \dots c_k} = \bar{r} + \sum_{j=1}^k b_{ic_j} \quad (7.2)$$

Here, the rating prediction is the same for all the users and depends only on the current context. Note that if the context is unknown, the model becomes equal to the item average. In addition, we considered a model that provides personalized, but non-context-aware predictions.

Figure 7.4 shows how the values for the learned parameters (b_{ic_j}) are distributed when the non-personalized model is computed (see Equation 7.2). For instance, there are approximately 70 parameters with a value between -0.5 and 0. The distribution of the parameter values is close to a gaussian distribution with average -0.23. Most of the contextual factors have small influence on the rating of a user and item pair. Moreover, the contextual factors that we have selected have a mixed effect on the rating prediction; some of them tend to increase the

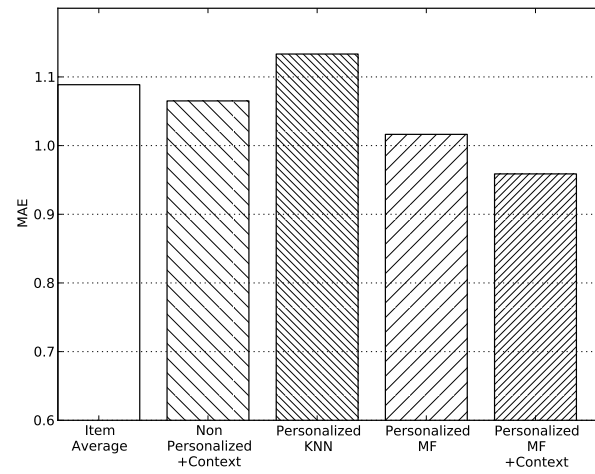
rating for an item and some of them not. But the majority of the contextual factors have a negative impact on the rating. This corresponds to the results reported in Section 7.4.

We estimated the performance of the considered models using repeated random sub-sampling validation. We did 100 splits into training and testing set with the training set containing 90% of the data. We recall that the rating data set consists of 20 POIs, 24 users, and 1484 ratings (as described in Section 7.4). We compared the proposed context-aware methods with the simple non-personalized prediction model given by the item average and with a user-based CF method [Herlocker et al., 2004] that is named “Personalized KNN” in all the Figures. We used the cosine angle between user rating vectors as user-to-user similarity.

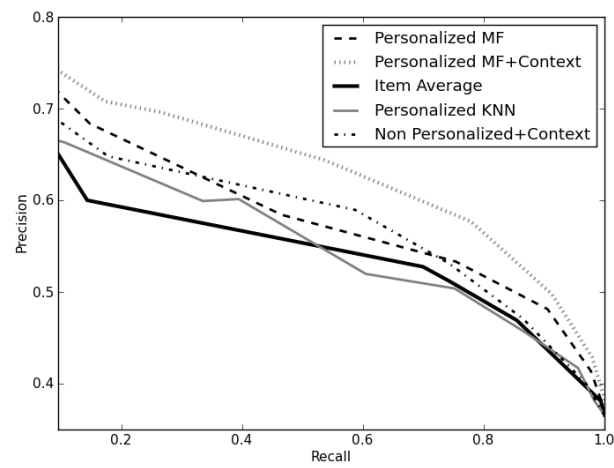
Figure 7.5(a) shows the Mean Absolute Error (MAE) of the models. The largest improvement with respect to the simple model based on the item average is achieved, as expected, by personalizing the recommendations (MF Personalized). This gives an improvement of 6.6%. This improvement is a bit smaller than what has been obtained in other cases comparing personalized vs. non-personalized rating predictions (e.g., [Anderson et al., 2003]). But we observe that a direct comparison of this result with other studies cannot be done since the benefit of personalization is strictly dependent on the quantity of the rating data, i.e., on the specific characteristics of the data set.

The personalized model can be further improved by contextualization producing an improvement of 11.9% with respect to the item average prediction (Personalized MF + Context), and a 5.6% improvement over the personalized model (MF Personalized). Note, that the non-personalized but contextualized predictions improve the accuracy of the item average prediction by 2.1% (non-personalized + Context). The user-based CF algorithm (Personalized KNN) has the worst MAE performance. This can be explained by the fact, that it can use only partial information from the training set, i.e., only the ratings that were gathered without any contextual information. Probably, better approaches could be identified. For instance, it might be that averaging all the ratings in the training set that a user gave to an item - in all the registered contextual conditions - could provide a better guess of the general rating of a user for an item irrespectively of the contextual condition. However, this is out of the scope of our study to design a new non-context-aware KNN algorithm for collaborative filtering using contextually tagged data.

In a recent paper [Esuli and Sebastiani, 2010] Esuli and Sebastiani stated that MAE is not a good measure for ordinal classification, such as rating prediction. Indeed, they observe that “an ordinal classifier that has classified all test items correctly aside from swapping equal numbers of items between two classes c_i and c_j , has perfectly estimated the distribution of items across the ordered classes, regardless of the number of swapped items and of the “distance” between c_i and



(a) MAE of different models



(b) Precision and Recall of different methods

Figure 7.5: Performance of different methods.

c_j ”. To make sure that we improve not only MAE but also ranking tasks we also measured the ranking performance of the proposed algorithms. Figure 7.5(b) shows Precision/Recall curves for all the considered methods. As it was proposed in [Herlocker et al., 2004] we considered items rated as 4 and 5 as relevant ones. To achieve different recall levels, we computed precision and recall when the system recommends items with a predicted rating higher than $\{1.1, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.2, 4.5, 4.8\}$. The clear winner for all the recall levels is our proposed personalized CF that takes into account contextual information. As usual in recommender systems, we are mainly interested in increasing precision, even if this will have a negative effect on recall. Assuming this, the second best method is CF based on matrix factorization followed by the proposed non-personalized method that takes context into account (non-personalized + Context). The worst performing methods, similarly to what was observed for MAE, are Item Average and user-based CF (Personalized KNN).

In conclusion, we can state that the proposed modeling approach can substantially improve the rating prediction accuracy when taking into account the contextual information.

The above described modeling approach was used also for generating explanations for the recommendations. Analyzing the learned parameters one can generate explanation based on the values of these parameters. More specifically, given an item i , user and contextual situation (c_1, \dots, c_k) , we identify the contextual factor j , among those specified in the contextual situation, with the highest impact, i.e., with the largest estimated parameter b_{ic_j} . Then, we generate a positive explanation for recommending item i using the contextual condition c_j . For example, if item i (let us say Castle Mareccio) is recommended in the contextual situation “temperature is cold and user is with children”, we observe if the parameter b_{ic_j} is greater than b_{ic_l} , assuming that index j refer to “temperature” and l refers to “companion” factors, and we explain that Castle Mareccio is recommended because it is cold. Other examples of explanations are presented in the next Section. Note that in other context-aware CF approaches such as the reduction-based approach [Adomavicius et al., 2005] there is no straightforward way to generate such explanations.

7.6 Context-Aware Place of Interest Recommendations

In this section we illustrate the main features of ReRex, a mobile context-aware recommender system. ReRex is an iPhone application that allows the user to obtain recommendations adapted to the recommendation context. The recommendations are computed by a server component that implements the predictive model described in the previous section. The iPhone application interacts with the server by means of a custom designed XML-based protocol;

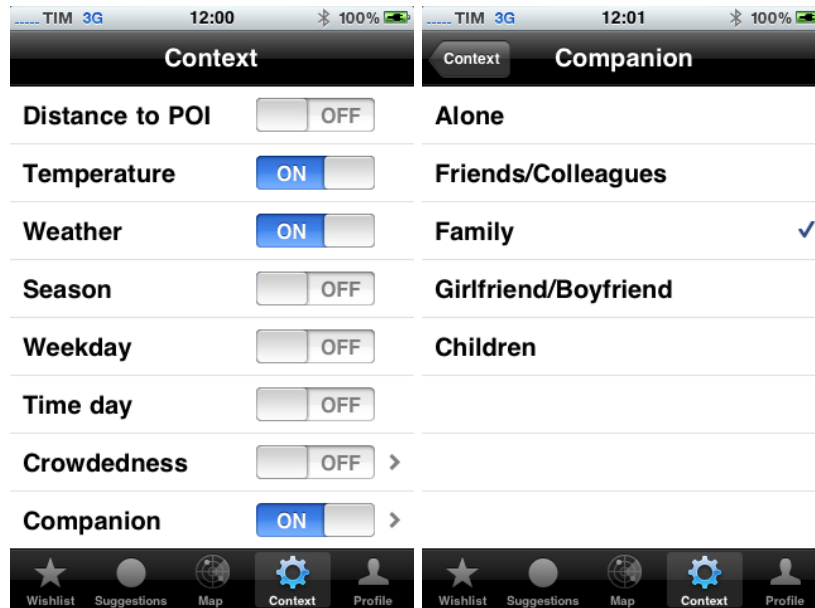


Figure 7.6: Context specification UI.

the client makes a recommendation request specifying contextual conditions and the server replies with a list of POI recommendations (including pictures and descriptions). The recommendations are determined by two types of information: the in-context ratings for POIs (provided by the data collection described in Section 7.4), and the average rating for the POIs given by the same user population. We will now describe this system illustrating a typical interaction.

In the initial step of the interaction with ReRex the user normally sets the context of the visit. Figure 7.6 shows the GUI for enabling and setting the values of the selected contextual factors. The user can switch on/off some of these factors, e.g., “Temperature” or “Weather”. When these factors are switched on the recommender system will take into account their current values (conditions) by querying an external weather forecast service. For other factors, the user is allowed to enter a value, for instance, the user can specify the group composition as in Figure 7.6 (right). The full set of contextual factors is the same as in the web application described earlier and their values could be found in Table 7.1. The contextual conditions: time of the day, day of the week, distance to POI, weather, season, and temperature are automatically obtained from other server components and are not entered by the user. The remaining contextual conditions, if the user has enabled them, must be entered manually by the user.

After the user has enabled some contextual factors and entered the contextual

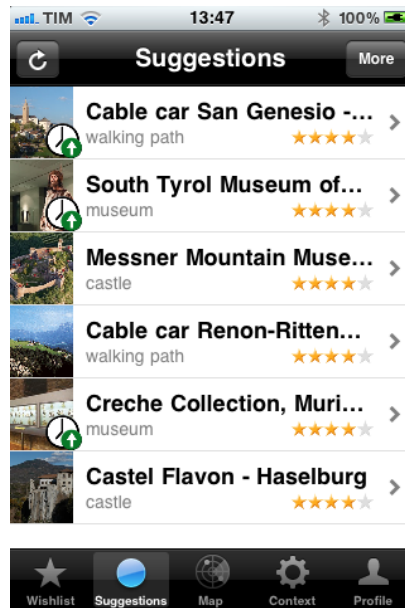


Figure 7.7: Suggestions GUI.

situation the system can be requested to provide recommendations. A short number of suggestions, namely six, are provided to the user as depicted in Figure 7.7. If the user is not happy with these suggestions he can click on the “More” button (upper right), and more suggestions are provided, after the user specified earlier the type of POIs he is looking for. In the suggestion visualization screen the user can touch any suggestion to have a better description of the POI as illustrated in Figure 7.8. It is worth noting that some of these suggestions, e.g. the top recommendation “Cable car San Genesio” are marked with an icon showing a small clock and a green arrow. These recommendations are particularly suited for the context. In the description of these recommendations there is an explanation like “This place is good to visit with family”. This is the contextual condition largely responsible for generating a high predicted rating for this item (as it was explained in the previous section).

ReRex offers some additional functionalities. Firstly, each recommended item can be saved in the user wish list. This is achieved by clicking on the star like icon with a green plus shown in the POI details screen (Figure 7.8). The wish list interface is similar in appearance to the suggestion list; it shows the selected suggestions and enables the user to browse their details. The user can also remove an items from the wish list. When a POI is visualized, as in Figure 7.8, the user can touch one of the buttons in the top part of the screen: Details, Map or



Figure 7.8: Details for a suggestion.

Feedback. Pressing Details provides more detailed information about the item. The Map button shows a Google map with the position of the POI and the current position of the user. Finally, if the user touches Feedback he is presented with a form where he can enter his context-dependent rating on the selected POI. We note that in the live user evaluation of ReRex, which is described later, we did not use the ratings entered by the users by means of this functionality. To compute the recommendations, we instead used the rating data set acquired with the web application described in Section 7.4, and the predictive model was computed off-line before the experiment took place. The rationale here is that during the evaluation experiment we wanted to keep the predictive model stable for all the users.

The last important feature of ReRex is related to its active behavior. If a contextual condition is changing, e.g., the temperature is getting cold, the user is notified in a simple way as shown in Figure 7.9. Moreover, it is worth noting that the user can at any moment enter new values for the context status and the suggestion list is updated accordingly. So, ReRex offers always new recommendations, and lets the user to explore the system and evaluate the quality of the recommendations for different values of the contextual variables.

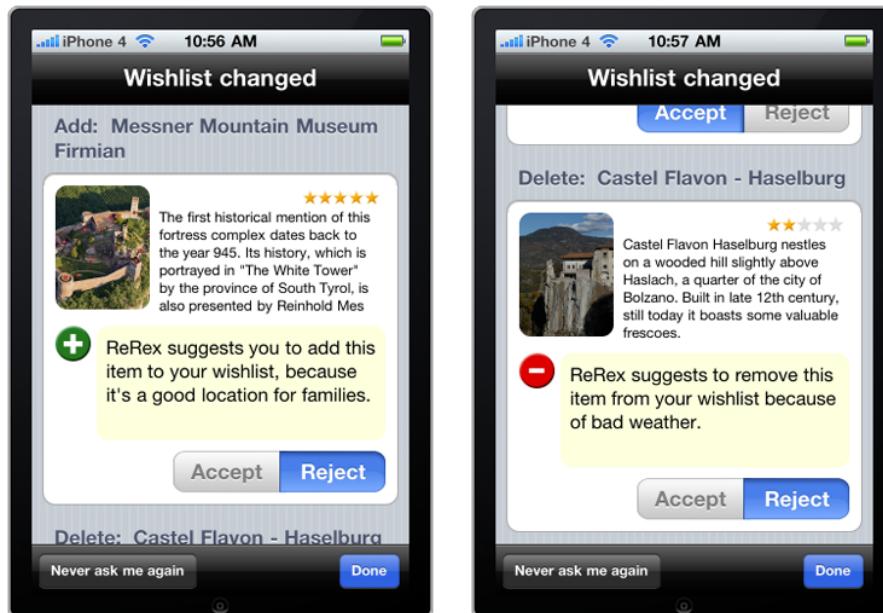


Figure 7.9: Notification of a recommendation change.

7.7 Evaluation

Previous assessments conducted on CARS were mainly based on off-line evaluations [Adomavicius et al., 2005], and the system effectiveness was measured as the precision in predicting correct recommendations, or as the accuracy in predicting the users's true ratings for items (as we did in Section 7.5). In these off-line evaluations the system predictions for a target user, either the recommendations or the ratings, are evaluated on a test set of items extracted from those already evaluated by the target user. So, in these cases the system is considered accurate if it correctly predicts the user's evaluation for a subset of items that the user actually rated (test set). To compute these predictions knowledge contained in a complementary subset of the ratings of the user (training set) is exploited [Shani and Gunawardana, 2010]. This type of evaluation, notwithstanding its large diffusion, is severely limited, as the test set, i.e., the items used to measure the system effectiveness, is a small fraction of all possible recommendations the system can make to the users. More importantly, the test set is not a random sample of these (future) recommendations. Hence, the performance of the deployed system actually could be rather different from the one observed in this type of analysis [Shani and Gunawardana, 2010, Marlin and Zemel, 2009].

For these reasons we have followed here also a second approach, and we

have measured the *subjective user satisfaction* for the system when the knowledge of some contextual variables is exploited: in the computation of the recommendations, and in the visualization of these suggestions to the user [Shani and Gunawardana, 2010]. We have measured the user satisfaction with a standard usability questionnaire [Lewis, 1995]. Moreover, in order to isolate the effect of the contextual factors on the user satisfaction we decided to generate the recommendations using a prediction model (as described in Section 7.5) that is not personalized, i.e., the recommendations are not dependent on the target user ratings. So, for instance the system does not exploit how the target user evaluates castles, churches, or markets neither in Bolzano nor elsewhere, before recommending some other POIs in Bolzano, and all the users will receive the same recommendations if they also entered the same contextual conditions⁵.

The recommender system that we have tested integrates two knowledge sources:

1. The general knowledge of a population of users that live in the city. Their ratings have been averaged to produce a popularity based ranking of what is more interesting and significant to suggest to a tourist. This corresponds to the average rating term in the model shown in Equation 7.2.
2. The overall effect of the context on the evaluation of the POIs. That knowledge is derived from the in-context ratings data of the users that tried the Web application described in Section 7.4. This knowledge is used to learn the remaining term in the model shown in Equation 7.2.

In other words, we implemented a recommender that knows which are the most important attractions of the city and exploits the knowledge of the contextual factors listed in Section 7.6 to adapt its suggestions to the specific context of the request and situation of the target user.

In order to measure the effectiveness of this approach we developed two variants of our ReRex iPhone mobile recommender system. The first is that described previously, and its rating prediction model is the non-personalized, but context-aware system described in Equation 7.2. The second variant is neither personalized nor context-aware, i.e., there is no possibility for the user to specify his current context, and the UI screens shown in Figure 7.6 were removed. The recommendations are sorted according to the item average rating, and are essentially those that the Tourist Office would make to a visitor without any special knowledge of the context of the visit. This is the non-personalized non-context-aware model that is mentioned in Section 7.5.2.

⁵We note that this is actually a big advantage for the proposed recommender, as it does not suffer from the “new user problem”, i.e., the impossibility to deliver a recommendation to a user new to the system, i.e., that has not entered yet any rating [Ricci et al., 2010].

In this way we could measure if the proposed context-aware prediction model, which is based on the in-context ratings collected with our methodology, can improve the user perceived relevance of the recommendations with respect to those generated by a non-context-aware model. We note that this is not a trivial question, since the in-context ratings, i.e., the influence of the context on the ratings, are those expressed by a generic population of users, and not those of the target user. Moreover, these evaluations and ratings are based on subjective evaluations of the POIs in contextual conditions that were only *imagined* by the users.

7.7.1 Usability Test

The recommender system was trained with the rating data (1664 ratings for 30 POIs) collected with the application described in Section 7.4. The test participants (20 users – working or studying at the Faculty of Computer Science in Bolzano) experimented with both variants of the system, in a random order, and executed, supported by each system, similar, but different tasks. For instance, in one of these two tasks the user was instructed to: *Imagine you are living in Bolzano. Suppose that it is a cold, rainy Wednesday and you are alone. Select a single point of interest of your choice and add it to your wish list. Then, suppose that your parents are coming to visit you. If needed, revise your previous selection and add another point of interest to your wish list.* A second similar, but different task was assigned when the user was asked to try the second variant; hence each user tried both variants and each variant was used to solve one (randomly assigned) of these tasks.

After the user completed the assigned task using one system, she was requested to fill out a usability questionnaire including the following statements:

Q1: It was simple to use this system.

Q2: The interface of this system is pleasant. We wanted to know how easily users can find their preferred points of interest using the system and get information about them.

Q3: The organization of information provided by the system is clear.

Q4: It was easy to find the information I needed.

Q5: The system is effective in helping me to complete the scenario.

Q6: It was easy to learn to use this system.

Q7: Overall, I am satisfied with this system.

Q8: I like using this system.

Q9: This system has all the functions and capabilities I expect it to have.

Q10: I am satisfied with the suggested points of interest.

Q11: I can effectively find interesting suggestions when using this system.

Q12: I understood the benefit of using the contextual conditions.

Q13: It was easy to specify the desired contextual conditions.

Q14: I am satisfied with the provided contextual explanations.

Q15: I believe that the contextual explanations are useful.

Q16: The contextual explanations provided by this system are clear.

Statements 12-16 were provided only in the questionnaire filled out after testing the context-aware version. The user could express a level of agreement to the statement ranging from from -2 (strongly disagree) to 2 (strongly agree). These questions were extracted, and slightly adapted to the scope of our investigation, from the IBM Computer System Usability Questionnaire [Lewis, 1995]. At the end of the interaction with the two variants we asked which system the user preferred (Q17) and which one suggested more appropriate points of interest (Q18).

The average response to the first eleven questions, which were asked after testing both interface variants, by the 20 users that evaluated the two system variants are shown in Figure 7.10. There is a clear preference of the users for the context-aware variant. Not all the observed differences are statistically significant; those significant are 5, 7, 9-11.

We measured the significance using parametric paired t-test and non parametric Wilcoxon Signed-Rank Test. Paired t-test assumes that differences of correlated responses follow the normal distribution. Even widely ignored, this assumption could be violated while using Likert scale. To confirm the significance we used Wilcoxon Signed-Rank Test that has no such assumptions about the distribution. We set alpha significance level equal to 0.05. Both tests agreed on the significance of the differences for Q5,7,9-11.

Hence we can observe that both systems are considered easy to use and their interface is nicely organized, but the context-aware system is:

- more effective in helping the user to complete the task (Q5);
- produces overall higher satisfaction (Q7);
- is more complete, in term of functionality (Q9);

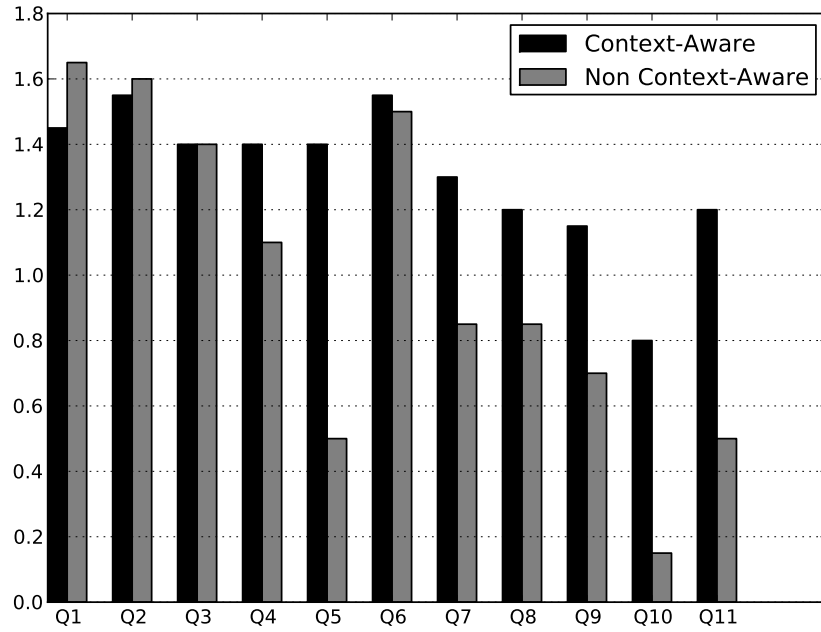


Figure 7.10: Usability analysis: average replies to the questions

- suggests POIs that make the user more satisfied (Q10);
- suggests more interesting POIs (Q11).

Regarding questions Q12-Q16, which were asked only after the context-aware variant was tested, the average results are: Q12 1,30; Q13 1,10; Q14 1,05; Q15 1,50; Q16 1,20. So, also with respect to the specific context-aware aspects the users were globally positive. We note that among these aspects the most interesting is the explanation support. Paradoxically the quality of the explanations scores low (Q14 1,05) but its importance scores high (Q15 1,50). This seems to indicate that the explanation is a very important component, and the user is particularly sensible to the quality of these explanation, but the formulation of these explanations can be surely further improved.

Finally, when the users were requested to directly compare the two variants (Q17 and Q18), 85% of the users *preferred the context-aware version*, while 95% of the users *considered the context-aware recommendations more appropriate*. In conclusion, although the number of testers was limited, this evaluation provided a clear evidence that the context-aware recommendations produced by the system

were more effective than those produced by the non-context-aware version, i.e., the recommendations normally provided to visitors of the city of Bolzano by the tourist office.

7.8 Conclusions and Discussion

In this chapter we have argued that the context is known to have a large impact on the user decision making and information systems' usage, but often the relationship between the context and the computer supported decision process is unknown and uncertain. Which contextual factor is relevant, in a specific decision making situation, is hard to predict and wrong assumptions may lead to unnecessary and misleading reasoning models.

Hence, in this chapter we have illustrated a methodology and three tools supporting its application. Two of them were used for acquiring data about the dependency of the users' evaluations for items on the context status, and the third for providing context-aware recommendations. The data collected with the first two tools have been used to build a mobile tourist assistant, ReRex, that suggests POIs to tourists according to the value of several contextual conditions. ReRex promotes (or pushes down) in a standard recommendation list the items that are more (or less) suitable for the particular context of the target user. In a live user experiment we have compared ReRex with a similar system, i.e., having the same user interface, but not adapting the recommendations to the context. We have observed a clear preference of the users for the context-aware variant and registered that the users consider the recommendations offered by the context-aware system more interesting and more relevant for their decision task. This indicates that the proposed methodology is able to deliver effective recommendations even if the modeled contextual dependency of the evaluations was not specific for the target user; they were actually acquired by asking the user to imagine the effect of context on their preferences, and even more interestingly the target user preferences, i.e., ratings on a set of items, were not exploited. So, the tested system was delivering non-personalized, but contextualized recommendations.

Best Context approach. In Section 7.3 and Section 7.4 we have showed a two step data acquisition approach. Here we want to note that we have proposed in [Baltrunas et al., 2010a] also an alternative approach for context-dependent data acquisition. We will briefly summarize that work here. In that approach for each (item, user) pair we acquired a subjective user evaluation of the most appropriate context for consuming the item in. We call this evaluation the “best context” for the item. Instead of asking to rate a particular item in several target contexts we ask the user to state under which contextual conditions she would prefer to consume an item. This is surely easier for the user, as it does not

require her to rate items in many different conditions. Such data can be used to answer a very important question: what items to recommend when the user is experiencing a particular context, for instance when the user is sad or is traveling by car. In fact, this can be easily solved by a process consisting of two steps: first, predicting the context that is best suited for any item, or at least for a large part of the items' catalogue, and then searching for the items whose predicted "best context" is exactly, or similar, to the target context of the query. The first step can be done off-line, and requires to predict for each (item,user) combination not a simple rating, as in classical Collaborative Filtering, but a context description. Only the second step should be performed on-line and can be easily computed as a k-nearest neighbor query over the best context descriptions of the items. We observe that the best context for an item should be predicted because it is unrealistic to assume that a user has already provided this evaluation for all the items in the catalogue. A set of these evaluations, on a subset of the items, can be used to predict the best context for the remaining items. Best context data acquisition approach is promising, but it needs a more careful investigation in the future. There are a number of open issues that still have to be explored. First of all we want to integrate the best context prediction with the classical rating prediction performed by collaborative filtering and build an hybrid approach. Then we would like to experimentally compare, in a live users experiment, if the context management functionality here proposed can generate more relevant recommendations compared with the rating in context data acquisition approach. For these reasons we consider this approach as a preliminary solution and we will dedicate some future work.

In conclusion, in this chapter, we have shown how the uncertain relationships between context and decisions can be explored and effectively used even if the data describing the relationship between contextual factors and item ratings were not as precise as those used by previous CARS systems. In fact, we have acquired the dependency between context and ratings by asking to a population of users to **imagine** the effect of the context on their evaluations. This is known to be different from the data measured by observing the real evaluations of the users in different context. But we have shown, notwithstanding these differences and therefore the limitations of the used data, that the final recommender could be perceived by the user as more effective than a non context-aware system. We have hence shown the practical advantage of the proposed approach. We stress that in our approach, we greatly reduce the cost of the context-dependent rating acquisition phase still providing useful and effective recommendations to the users.

We observe that we are now applying the proposed approach in a different decision making scenario, namely music recommendation for a group of passengers in a car, to understand to what extent the approach can be generalized to other tasks. Moreover, as future work we plan to conduct another experiment where

a recommender that does exploit the ratings of the target user to produce the recommendations is compared to the same recommender that we have illustrated here, i.e., a recommender that uses only the contextual situation of the user and exploits a general model, i.e., valid for any user, of the influence of the context on the items' evaluations. We want to understand how important is to adapt the recommendations to the context, compared to the classical adaptation performed by recommender systems, i.e., that based on the ratings of the user on a set of items.

Part IV

Conclusions and Summary

Conclusions and Future work

8.1 Summary and Conclusions

In this thesis we have analyzed how **context** can be exploited to improve Collaborative Filtering algorithms. Here we adopted the definition of context as “... any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application.” [Dey, 2001]. In a Recommender System context could be the user’s companion, the weather, the temperature or the location of the user when she is requesting a recommendation for a place to visit.

This research is motivated by the increasing need of information discovery services that in some domains are replacing classical search methods. On-line users often know only their general interests and want to discover the best items (information, services, or products) for their current state. These are usually items that the user did not know that existed in the first place. Recommender Systems emerged because they can offer solutions for this information discovery needs. These are software tools and techniques suggesting to a user a well selected set of items matching the user’s taste and preferences. The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read. They are widely used in Web-based e-commerce applications to help online users to choose the most suitable products, e.g. movies, CDs, books, or travels.

Classical Recommender Systems do not take into account contextual information, but context can help to better understand the information needs of a user and therefore to generate more useful recommendations. For example, in a tourism related application the amount of time that the user has for a visit would not be taken into account by a classical recommender system. But it is evident that this simple contextual information can make a big difference in the

type and quality of the recommendations.

In this thesis we have analyzed how such contextual information could be leveraged, in a precise and efficient way, to generate more relevant recommendations. The proposed techniques exploit historical behavioral data where user ratings are labelled with the context they were experienced in. We built our proposed approach on state of the art solutions and extended them in many ways.

In the first part of the thesis, we introduced and evaluated a methodology supporting the *full cycle* of building a CACF system. It includes: two phases of data acquisition, a well tuned algorithm capable of generating explanations for the recommendations, and a graphical user interface. We evaluated each step of this process offline and we conducted a user study to show how contextual information can increase user satisfaction with the generated recommendations. In the first data collection step we propose to assess the importance of contextual factor, for an item category. In particular, we considered Places of Interest (POI), and we assessed how much a contextual factor, such as weather, influences the user decision to choose a POI. We found out that there are some contextual factors that indeed seem to be relevant for all the categories; in that scenario these are *distance*, *time available*, and others. While, others often appear to be less relevant: *transport*, *travel goal*, *day of the week*. Next, we collected rating in context. Here, a population of users indicated how a contextual factor influences their decisions to visit a POI, and we compare this with the default interest in visiting it. We saw that there is a statistically significant difference between ratings in context and “default” ratings, i.e., evaluations provided without any specific context influencing them. We built a prediction model that exploits such differences, and provides more accurate rating predictions. At the end we built and evaluated a mobile application (iPhone) that is able to offer such recommendations to a users and to explain them.

In the second part of this thesis, we concentrated on investigating algorithms for CACF. We created more scalable and more accurate algorithms compared to those previously known in literature. These new algorithms can be used with larger data sets, having also more contextual dimensions. One of the proposed algorithms is item-splitting. This splits the available ratings of an item into two new virtual items if there is a statistically significant difference between the ratings of these two new items. It was showed to scale better and to be more accurate compared to the classical reduction based approach. More recently we investigated a Tensor Factorization approach for CACF that employs a pure model based technique to CACF. It was showed to be more accurate than any other method previously tested. We also investigated new ways to incorporate additional information into the user-to-user similarity measure of CF. We considered item weighting and item selection approaches and we proved that these techniques can outperform standard user-based CF.

Additionally, we analyzed the “companion” contextual dimension as a special

case. Having rich user profiles of the companion(s) we were able to build accurate recommendations for groups. We have shown that sometimes these recommendations can be even more accurate than the ones build for a single person. Especially, if the person has a small profile and he is similar to the other group members.

8.2 Future Work

When me and my colleagues started our Ph.D. cycle we had a very helpful BIT seminar series for graduate students by prof. Fitzgerald. He made an overview of the life of a Ph.D. student and gave a perspective on how to write a thesis. One of his ideas, which now I can understand better, was that “a Ph.D Thesis can not be finished. It is abandoned.” Therefore, in this section we list and motivate several promising topics for future research.

Implicit data domains. The recent research in the Recommender Systems has focused in generating recommendations based on explicit user feedback data such as ratings (e.g. Netflix prize data). However, the large majority of the data that is available and easy (cheap) to collect in online systems is actually implicit. Online one can collect user clicks, purchases, installations of applications, etc. These events can only provide an indirect indication of the user preferences for items, i.e., the visited page, or the purchased product. There are many other situations where implicit feedback are easy to collect, but not explicit ratings. For instance, a recommender system running on the user mobile phone or in the car entertainment system should favor implicit over explicit feedback to avoid the additional effort for the user and the negative impact of task disruption [Karlson et al., 2010]. Implicit feedback types include purchase and browsing history, usage history, search patterns, or even mouse movements. Although implicit feedback data are in some sense similar to explicit feedback data e.g. they can be represented as *user, item* tuples in a two-dimensional matrix format, there are some notable differences that make modeling implicit data a unique challenge.

Context plays a vital role in implicit feedback data sets. When giving an explicit feedback, the user can expect that his rating will be further used for personalization. Therefore, the user could choose not to rate a book that was bought as a gift. In the systems that are based on implicit feedback the users control over the feedback is lost. The recommendation system gathers all the user’s activity and it is up to the system to decide which information is relevant for the recommendations and in which situation to exploit it. Additional information gathered together with implicit user feedback could partially solve this issue. The system could automatically discover the context and provide

relevant recommendations for the specific context. In this sense, context can be seen as a query refinement, i.e., a CARS tries to retrieve the most relevant items for a user, given the knowledge of the context of the request.

Evaluating alternative data acquisition techniques. To bootstrap our CACF system, we built a two phase data acquisition tool. It is clear that even with such techniques context-dependent rating acquisition is very expensive. We started investigating a novel approach for collecting and using context data in recommender systems. In the new approach for each (item, user) pair we acquire a subjective user evaluation of the most appropriate context for consuming the item in. We called this evaluation the “best context” for the item. Instead of asking user to rate a particular music track in several target contexts we asked the user to state under which contextual conditions she would prefer to consume an item. This is surely easier for the user, as it does not require her to rate items in many different conditions. This results in an association between a user, an item and a context. We already analyzed several methods for making accurate “best context” predictions and found one that outperforms other methods. However, there are a number of open issues that still have to be explored. First of all we want to integrate the best context prediction with the classical rating prediction performed by collaborative filtering and build a hybrid approach. Then we would like to experimentally compare, in a live users experiment, if the new context management functionality can generate more relevant recommendations.

Analyzing additional methods for context-dependent explanations. In Chapter 7 we presented a prototype system that is able to make context-aware recommendations. It uses a rating prediction method together with a heuristic that is used to explain these recommendations. Additional research should be done in order to understand in which situations and for what purposes these explanations should be provided. Would it make interaction with system more fruitful and pervasive? Would the user accept the explained recommendations more than others? Moreover, the current explanation heuristic highly depends on the underlying prediction algorithm. This clearly limits the diffusion of such an explanation algorithms. It would be interesting to investigate methods that are able to generate explanations accessing the rating prediction component as a black box. This would allow us to use such systems with any kind of prediction engine.

Sequential Recommendations. In most of the CACF approaches context is treated as a feature of the rating. This usually does not include recent history of user consumption [Adomavicius et al., 2005, Baltrunas and Ricci, 2009a, Karatzoglou et al., 2010]. Clearly this is a very important contextual dimension

and needs a special treatment. Sequential recommendations are important in many domains. For example, while making recommendations for places of interest, we should take into account what did user visit till that moment. Such information could make recommendations more diverse and interesting. In music domain, recently played songs should be taken into account in order to make transitions between tracks smoother and to give more internal coherence to the generated playlist. We are planning to employ reinforcement learning as a meta recommender in order to solve these issues. A high level recommendation strategy, i.e., the selection of the category of items that should be recommended next, could be based on the recent items, and the precise recommendations, within that category, could be done using the already analyzed CACF techniques.

Actively learning current context. In many cases contextual information could be automatically retrieved using sensors. This includes weather, temperature and location. However, for some contextual conditions such as companion, user should actively provide the contextual information. On one hand, knowing such information could improve the recommendation accuracy. On the other hand, the user needs to make additional work in order to provide such information. Clearly, this additional input should be rewarded with better recommendations or more fun and engaging interaction. We should study when it is worth to actively ask user for explicit input and when it is better to make a best guess from available information.

In conclusion, context aware recommender systems are now starting to gain more and more popularity. Initially most of the researchers strived to prove that CARS could outperform context-free approaches. Now we are looking forward to new ideas about how to exploit contextual information not only for increasing the prediction accuracy of the recommender. We believe that a deeper investigation of this topic could lead to more intelligent and useful Recommender Systems. The research field is rich of real world problems that are challenging and call for more investigation.

Symbol Table

User	$u \in U$
Item	$i \in I$
Number of users	m
Number of items	n
Ratings (matrix, tensor or set)	R
Permutation of the items I , representing the ranked list recommendations for user u	σ_u
Rating given by user u , for item i in context c	r_{uic}
Context	$C = C_1 \times C_2, \times \dots \times C_k$
Contextual dimension (aka contextual factor)	$C_j = \{v_{j1}, v_{j2}, \dots, v_{jz}\}$, where z is the number of states (values) the contextual dimension C_j can take.
Contextual condition (aka contextual value)	$v_{j1} \in C_j$
MAE	Mean Absolute Error. Defined as the mean error between the rating prediction and the real rating given by a user.

Table A.1: Symbol Table

Ranking of Contextual Factors

In phase 1 of our study on the relevance of context factors (see Sect. 7.3) we measured the relevance of a context factor by the normalized mutual information between the response of the user and each context factor: the higher the mutual information the better the context factor can explain the response of the user to the questions in the interviews. In the Table B.1, we present an overview of the context factors ordered by different POI category:

castle	church or monastery	cycling or mountain biking	folk festival, arts and crafts event	museum
distance knowledge about area time available season day week crowdedness day time mood companion travel goal transport temperature weather budget	distance time available mood day time transport travel goal temperature companion weather crowdedness season knowledge about area budget day week	budget time available crowdedness season weather temperature mood knowledge about area day time transport companion travel goal day week distance	distance temperature knowledge about area weather time available companion season budget day time crowdedness day week travel goal transport mood	distance budget knowledge about area temperature time available companion weather crowdedness travel goal season day week day time transport mood
music event	nature wonder	spa	theater event	walking
crowdedness day week time available mood companion distance day time budget transport temperature travel goal season knowledge about area weather	distance day week temperature crowdedness season time available weather companion day time mood travel goal transport budget knowledge about area	distance knowledge about area crowdedness season time available weather temperature companion travel goal day time mood budget transport day week	time available day time distance budget temperature knowledge about area day week mood travel goal season crowdedness companion weather transport	distance budget temperature crowdedness knowledge about area time available weather season day time mood day week transport travel goal companion

Table B.1: Ranking of context factors to their association with the user responses on the influence of a factor on their decision to visit an item.

context value	dimension	no. ratings	p-value	MCN	MCY	Effect
Castle						
far away	distance	15	0.000408107	3.802857143	2.466666667	↓
winter	season	11	0.022165228	3.80754717	2.636363636	↓
cold	temperature	7	0.06437363	3.897590361	2.857142857	↓
Church or Monastery						
night time	day-time	5	0.045900071	2.583333333	1.2	↓
far away	distance	6	0.055531068	2.35	1.166666667	↓
Museum						
sad	mood	14	5.11E-05	2.789655172	1.642857143	↓
activity/sport	travel-goal	6	0.000734366	2.640625	1.333333333	↓
active	mood	9	0.000864708	2.644067797	1.444444444	↓
far away	distance	25	0.001561059	2.776493256	1.92	↓
lazy	mood	6	0.011122893	2.9921875	1.833333333	↓
half day	time-available	15	0.01491846	2.731958763	1.933333333	↓
warm	temperature	17	0.015995573	2.701949861	2.058823529	↓
night time	day-time	15	0.023894116	2.933753943	2	↓
clear sky	weather	10	0.029247374	3.085714286	2.3	↓
rainy	weather	17	0.034039257	3	3.764705882	↑
with friends/colleagues	companion	5	0.035327383	2.614457831	1.4	↓
visiting friends	travel-goal	6	0.036527638	2.694915254	1.833333333	↓
budget traveler	budget	10	0.038209253	3.093457944	2.1	↓
hot	temperature	10	0.050943446	3.168224299	2	↓
weekend	day-week	14	0.057615252	2.593333333	2.071428571	↓
morning time	day-time	10	0.070290314	2.796116505	1.9	↓
snowing	weather	6	0.092024039	3.328358209	4.166666667	↑
Walking Path						
night time	day-time	7	4.01E-79	3.78343949	1	↓
far away	distance	13	1.10E-05	3.863157895	2.384615385	↓
cold	temperature	8	0.000243597	3.8	1.875	↓
winter	season	6	0.000274175	3.9140625	2.333333333	↓
with friends/colleagues	companion	6	0.000849203	3.850746269	4.833333333	↑
crowded	crowdedness	8	0.002333888	3.879310345	2.75	↓
working day	day-week	12	0.00238071	3.943548387	2.75	↓
half day	time-available	5	0.003014991	4.00990099	1.6	↓
more than a day	time-available	5	0.006695087	3.891891892	4.8	↑
lazy	mood	7	0.008503253	4.02919708	4.714285714	↑
alone	companion	6	0.091310886	3.830769231	2.833333333	↓

Table B.2: Average rating of the POIs of a certain category.

B.1 How Does Context Influence the Decisions of Users?

The table in this section presents a comparison between ratings of POI in Bolzano without any context and ratings of the same items assuming a certain context value to hold. In order to keep the sampling of the necessary probability distributions by means of the user study in Sect. 7.4 tractable and to stay in line with the linear predication model for collaborative filtering we assume context values to be independent:

$$P(R|C_1, \dots, C_k) \approx \prod_{i=1}^k P(R|C_i)$$

In order to find out which context values C_i were considered as relevant for their context-dependent rating of POI we compare $P(R|C_i)$ to $P(R)$ with the help of a t -test. In Table B.2 we show the context values that induce a statistically significant difference on the average rating of the POIs of a certain category.

Data Sets

C.1 Data Sets Used in Off-Line Evaluation

We tested the proposed method on several real world and a number of semi-synthetic context-dependent data sets. We want to thank the Yahoo!, Grouplens, Adomavicius et al. and Hideki Asoh et al. who made their data sets available for our research purposes. The summary of the data sets is provided in Table C.1.

Name	Entries	Users	Items	Context. Dims.	Sparsity
Adom	1464	84	192	5	96%
Food	6360	212	20	2	84.4%
Yahoo!	221K	7.6K	11.9K	2	99.7%
Yahoo! Semi-Synthetic	221K	7.6K	11.9K	2	99.7%
Movielens	100K	943	1682		96%
Bolzano POIs	1484	24	20	14	

Table C.1: Summary of the data sets.

Here sparsity is given as a ratio of number of ratings in the $user \times item$ matrix to the total cells in the matrix, and it is expressed in percentage. For context-aware data sets this definition could be confusing because these data sets can have several ratings for the same $user \times item$ pair. For example in Bolzano POIs data set we have more entries in the data set than $user \times item$ pairs in the matrix.

C.1.1 Adom

In many of our experiments we used the data set provided by Adomavicius et al. [Adomavicius et al., 2005]. We removed the entries without a full description

of the contextual variables. At the end we obtained a data set containing 1464 ratings by 84 users for 192 movies (hence a rather small data set compared with other standard data set used in CF research). The ratings were collected by asking college students to fill out a questionnaire on movies using a rating scale ranging from 1 to 13 (absolutely loved) and additional information about the the context of the movie watching experience. In other words, the authors, in addition to the ratings, recorded if the movie was watched on weekend or weekday, in a movie theater or at home, and with whom did they watched it. Moreover, the system registered additional information that was not used in [Adomavicius et al., 2005] such as if the user would recommend this movie to a friend. In our experiments we used 5 contextual features: a)companion {friends, parents, girlfriend, alone, colleagues}, b)day of the week {weekday, weekend, don't remember}, c)if it was opening weekend {yes, no, don't remember}, d)would the user recommend it to a friend {Excellent, Good, Fair, Bad}, e) year seen {2000,2001,2002}.

We note that the data set and its features are slightly different from the original experiment [Adomavicius et al., 2005]. Our obtained data set had more ratings and took into account features that were not considered before, i.e., the year when the movie was seen and if the user would recommend this movie to a friend. These differences were caused by difficulties to restore the original data set from the back-up and to understand what feature value corresponds to which question in the on-line questionnaire. We state this only to warn the reader to not compare the performance measures we report here with those included in [Adomavicius et al., 2005].

C.1.2 Food

Another real world data set was gathered, analyzed and kindly provided by Hideki Asoh *et al.* [Ono et al., 2009]. It contains food rating data from 212 users on 20 food menus. To acquire the data, the authors designed a two stage Internet questionnaire survey. The users were asked to rate the food menu while being in different levels of hunger. Moreover, some ratings were done while really experiencing the situation (i.e., the participants were hungry and ordered the menu) and some while imagining the situation. For example, in the imagined situation participants could be full, but they were requested to provide a rating for a food menu imagining that they were hungry. For this data set we use two context features: the three degrees of hunger and binary feature specifying if the situation was virtual or real. For more details on the data set we refer the reader to [Ono et al., 2009].

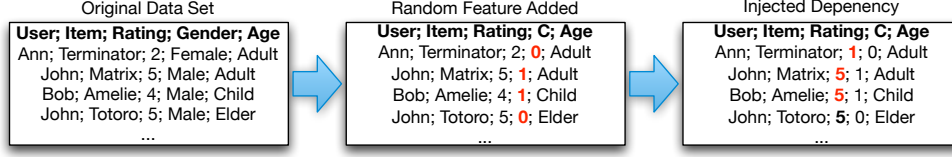


Figure C.1: Example of generating semi-synthetic data set.

C.1.3 Yahoo! Webscope

The third real world data set that we used was provided by [Yahoo, 2007] Yahoo! Webscope research project. It is much bigger than the data sets described above and contains 221K movie ratings with scale $\{1, 2, 3, 4, 5\}$, for 11,915 movies by 7,642 users. The Yahoo! data set contains user age and gender features. We used 3 age groups: users below 18 (u18), between 18 and 50 (18to50), and above 50 (a50). The gender and age features played the role of a contextual variable in CACF algorithms.

C.1.4 Yahoo! Semi-Synthetic Data Sets

In order to control the influence of the contextual features and analyze various properties of the CACF algorithms we also generated some semi synthetic data sets. These were generated using the original Yahoo! data set. We modified the original Yahoo! data set by replacing the gender feature with a new artificial feature $c \in \{0, 1\}$ that was assigned randomly to the value 1 or 0 for each rating. This feature c is representing a contextual condition (uniformly distributed) that could affect the rating. Then, we randomly chose α fraction of items from the data set and then among these items we randomly chose β fraction of their ratings. For all these ratings, we increased (decreased) the rating value by one if $c = 1$ ($c = 0$) and if the rating value was not already 5 (1). For example, if $\alpha = 0.9$ and $\beta = 0.5$ the corresponding synthetic data set has 90% the items' profiles altered, and for each of these altered items 50% of their ratings were altered. We generated various semi-synthetic data sets varying α and β parameters. Thus, in these data set the contextual condition is more “influencing” the rating value as α and β increase.

The schema of the generation procedure of the artificial data can be seen in Figure C.1. To keep it simple, in this figure we show the data as a list of tuples with gender and age feature. Bold red font indicate changes from the last step of the data set generation. The leftmost box shows the initial Yahoo! data set, containing user ratings for the items, age and gender information. In the middle box we show how the gender feature is replaced with the contextual feature c .

The values (1 or 0) are then assigned randomly to each tuple. The rightmost box shows the final step of injecting the dependencies. We choose a α fraction of items and modify β fraction of ratings of those items. In that example, the rating of movie “Totoro” was not changed, whereas the ratings of all the other tuples were modified according to the feature c as described above.

C.1.5 Movielens

The MovieLens [MovieLens, 2007] data set contains 100K ratings with scale $\{1, 2, 3, 4, 5\}$, for 1682 movies by 943 users, who have rated 20 and more items. The data sparsity of this dataset is 96%. This is the standard data set for off-line evaluation in the CF research literature.

C.1.6 Bolzano POIs

Bolzano POIs data set was gathered by our research team as a part of ReRex project, which is described in Chapter 7. It contains 1484 ratings with scale $\{1, 2, 3, 4, 5\}$ by 24 users for 20 items. In an Web-Interview a single POI is considered, and four ratings are requested: first, in general, how likely is the user to visit the POI, and then the same evaluation but assuming that three alternative contextual conditions are given. Please see Chapter 7 for the full details.

Bibliography

- [Abowd et al., 1997] Abowd, G., Atkeson, C., Hong, J., Long, S., Kooper, R., and Pinkerton, M. (1997). Cyberguide: A mobile context-aware tour guide. *ACM Wireless Networks*, 3(5):421–433.
- [Adomavicius and Kwon, 2007] Adomavicius, G. and Kwon, Y. (2007). New recommendation techniques for multicriteria rating systems. *IEEE Intelligent Systems*, 22(3):48–55.
- [Adomavicius et al., 2005] Adomavicius, G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145.
- [Adomavicius and Tuzhilin, 2001] Adomavicius, G. and Tuzhilin, A. (2001). Multidimensional recommender systems: A data warehousing approach. In Fiege, L., Mühl, G., and Wilhelm, U. G., editors, *WELCOM*, volume 2232 of *Lecture Notes in Computer Science*, pages 180–192. Springer.
- [Adomavicius and Tuzhilin, 2005] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749.
- [Adomavicius and Tuzhilin, 2008] Adomavicius, G. and Tuzhilin, A. (2008). Context-aware recommender systems. In Pu, P., Bridge, D. G., Mobasher, B., and Ricci, F., editors, *RecSys*, pages 335–336. ACM.
- [Adomavicius and Tuzhilin, 2010] Adomavicius, G. and Tuzhilin, A. (2010). Context-aware recommender systems. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*, pages 217 – 250. Springer.

- [Aha, 1998] Aha, D. W. (1998). Feature weighting for lazy learning algorithms. In *Feature extraction, construction and selection : a data mining perspective*, volume SECS 453, pages 13–32. Kluwer Academic, Boston.
- [Ahn et al., 2006] Ahn, H., Kim, K.-j., and Han, I. (2006). Mobile advertisement recommender system using collaborative filtering: Mar-cf. In *KGSM-Conference Papers*. The Korea Society of Management Information Systems.
- [Amatrian et al., 2010] Amatrian, X., Jaimes, A., Oliver, N., and Puriol, J. M. (2010). Data mining methods for recommender systems. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P., editors, *Recommender Systems Handbook*, pages 39–71. Springer Verlag.
- [Anand and Mobasher, 2006] Anand, S. S. and Mobasher, B. (2006). Contextual recommendation. In Berendt, B., Hotho, A., Mladenic, D., and Semeraro, G., editors, *WebMine*, volume 4737 of *Lecture Notes in Computer Science*, pages 142–160. Springer.
- [Anderson et al., 2003] Anderson, M., Ball, M., Boley, H., Greene, S., Howse, N., Lemire, D., and McGrath, S. (2003). Racofi: A rule-applying collaborative filtering system. In *Proceedings of COLA03*. IEEE/WIC.
- [Arrow, 1950] Arrow, K. J. (1950). A difficulty in the concept of social welfare. In *Journal of Political Economy* 58(4), pages 328–346. The University of Chicago Press.
- [Asoh et al., 2010] Asoh, H., Motomura, Y., and Ono, C. (2010). An analysis of differences between preferences in real and supposed contexts. In *Proceedings of the first Workshop on Context-Aware Recommender Systems*.
- [Baltrunas and Amatriain, 2009] Baltrunas, L. and Amatriain, X. (2009). Towards time-dependant recommendation based on implicit feedback. In Adomavicius, G. and Ricci, F., editors, *Workshop on Context-Aware Recommender Systems*.
- [Baltrunas et al., 2010a] Baltrunas, L., Kaminskas, M., Ricci, F., Rokach, L., Shapira, B., and Luke, K.-H. (2010a). Best usage context prediction for music tracks. In *2nd Workshop on Context-Aware Recommender Systems*.
- [Baltrunas et al., 2011a] Baltrunas, L., Ludwig, B., Peer, S., and Ricci, F. (2011a). Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing*, pages 1–20. 10.1007/s00779-011-0417-x.

- [Baltrunas et al., 2011b] Baltrunas, L., Ludwig, B., and Ricci, F. (2011b). Context relevance assessment for recommender systems. In Pu, P., Pazzani, M. J., André, E., and Riecken, D., editors, *IUI*, pages 287–290. ACM.
- [Baltrunas et al., 2010b] Baltrunas, L., Makcinskas, T., and Ricci, F. (2010b). Group recommendations with rank aggregation and collaborative filtering. In *RecSys '10: Proceedings of the fourth ACM conference on Recommender systems*, pages 119–126, New York, NY, USA. ACM.
- [Baltrunas and Ricci, 2007] Baltrunas, L. and Ricci, F. (2007). Dynamic item weighting and selection for collaborative filtering. In Berendt, B., Mladenic, D., Semeraro, G., Spiliopoulou, M., Stumme, G., Svatek, V., and Zelezny, F., editors, *Web Mining 2.0 International Workshop located at the ECML/PKDD 2007*, pages 135–146.
- [Baltrunas and Ricci, 2008] Baltrunas, L. and Ricci, F. (2008). Locally adaptive neighborhood selection for collaborative filtering recommendations. In Nejdl, W., Kay, J., Pu, P., and Herder, E., editors, *AH*, volume 5149 of *Lecture Notes in Computer Science*, pages 22–31. Springer.
- [Baltrunas and Ricci, 2009a] Baltrunas, L. and Ricci, F. (2009a). Context-based splitting of item ratings in collaborative filtering. In Bergman, L. D., Tuzhilin, A., Burke, R. D., Felfernig, A., and Schmidt-Thieme, L., editors, *RecSys*, pages 245–248. ACM.
- [Baltrunas and Ricci, 2009b] Baltrunas, L. and Ricci, F. (2009b). Context-dependent items generation in collaborative filtering. In Adomavicius, G. and Ricci, F., editors, *Proceedings of the 2009 Workshop on Context-Aware Recommender Systems*.
- [Baltrunas and Ricci, 2009c] Baltrunas, L. and Ricci, F. (2009c). Item weighting techniques for collaborative filtering. In Berendt, B., de Gemmis, M., Semeraro, G., Spiliopoulou, M., Stumme, G., Svatek, V., and Zelezny, F., editors, *Knowledge Discovery Enhanced with Semantic and Social Information*, volume 220 of *Studies in Computational Intelligence*, pages 109–127. Springer Berlin / Heidelberg.
- [Baltrunas and Ricci, 2010] Baltrunas, L. and Ricci, F. (2010). Context-dependent recommendations with items splitting. In Melucci, M., Mizzaro, S., and Pasi, G., editors, *IIR*, volume 560 of *CEUR Workshop Proceedings*, pages 71–75. CEUR-WS.org.
- [Basilico and Hofmann, 2004] Basilico, J. and Hofmann, T. (2004). Unifying collaborative and content-based filtering. In *Proc. Intl. Conf. Machine Learning*, pages 65–72, New York, NY. ACM Press.

- [Bazire and Brézillon, 2005] Bazire, M. and Brézillon, P. (2005). Understanding context before using it. In *Modeling and Using Context*, pages 29–40. Springer.
- [Bennett et al., 2007] Bennett, J., Lanning, S., and Netflix (2007). The Netflix prize. In *KDD Cup and Workshop in conjunction with KDD*.
- [Berkovsky et al., 2007] Berkovsky, S., Kuflik, T., and Ricci, F. (2007). Cross-domain mediation in collaborative filtering. In Conati, C., McCoy, K. F., and Paliouras, G., editors, *User Modeling*, pages 355–359. Springer.
- [Berkovsky et al., 2010] Berkovsky, S., Luca, E. W. D., Said, A., and Hermanns, J., editors (2010). *The Challenge on Context-aware Movie Recommendation (CAMRa2010)*.
- [Billsus and Pazzani, 1998] Billsus, D. and Pazzani, M. J. (1998). Learning collaborative information filters. In Shavlik, J. W., editor, *ICML*, pages 46–54. Morgan Kaufmann.
- [Breese et al., 1998] Breese, J. S., Heckerman, D., and Kadie, C. M. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In Cooper, G. F. and Moral, S., editors, *UAI*, pages 43–52. Morgan Kaufmann.
- [Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A.
- [Buriano et al., 2006] Buriano, L., Marchetti, M., Carmagnola, F., Cena, F., Gena, C., and Torre, I. (2006). The role of ontologies in context-aware recommender systems. In *MDM 06: Proceedings of the 7th International Conference on Mobile Data Management (MDM06)*, page 80, Washington, DC, USA. IEEE Computer Society.
- [Burke, 2002] Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370.
- [Carolis et al., 2009] Carolis, B. D., Mazzotta, I., Novielli, N., and Silvestri, V. (2009). Using common sense in providing personalized recommendations in the tourism domain. In Adomavicius, G. and Ricci, F., editors, *Proceedings of the 2009 Workshop on Context-Aware Recommender Systems*.
- [Chen, 2005] Chen, A. (2005). Context-aware collaborative filtering system: Predicting the user’s preference in the ubiquitous computing environment. In Strang, T. and Linnhoff-Popien, C., editors, *Location- and Context-Awareness*, volume 3479 of *Lecture Notes in Computer Science*, pages 244–253. Springer Berlin / Heidelberg.

- [Chen and Kotz, 2000] Chen, G. and Kotz, D. (2000). A survey of context-aware mobile computing research. Technical report, Dartmouth College Hanover, NH, USA, Hanover, NH, USA.
- [Coppersmith et al., 2006] Coppersmith, D., Fleischer, L., and Rudra, A. (2006). Ordering by weighted number of wins gives a good ranking for weighted tournaments. In *SODA 06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithms*, pages 776–782, New York, NY, USA. ACM.
- [Crossen et al., 2002] Crossen, A., Budzik, J., and Hammond, K. J. (2002). Flytrap: intelligent group music recommendation. In *IUI 02: Proceedings of the 7th international conference on Intelligent User Interfaces*, pages 184–185, New York, NY, USA. ACM.
- [Deerwester et al., 1990] Deerwester, S. C., Dumais, S. T., Landauer, T. K., Furnas, G. W., and Harshman, R. A. (1990). Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.
- [Dey et al., 2001] Dey, A., Salber, D., and Abowd, G. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, Volume 16 (2-4),:97–166.
- [Dey, 2001] Dey, A. K. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7.
- [Domingues et al., 2009] Domingues, M., Jorge, A. M., and Soares, C. (2009). Using contextual information as virtual items on top-n recommender systems. In Adomavicius, G. and Ricci, F., editors, *Workshop on Context-Aware Recommender Systems*.
- [Dourish, 2004] Dourish, P. (2004). What We Talk About When We Talk About Context. *Personal and Ubiquitous Computing*, 8.
- [Dunlop et al., 2004] Dunlop, M. D., Morrison, A., McCallum, S., Ptaskinski, P., Risbey, C., and Stewart, F. (2004). Focussed palmtop information access combining starfield displays and profile-based recommendations. In Crestani, F., Jones, M., and Mizzaro, S., editors, *Proceedings of workshop on Mobile and Ubiquitous Information Access, LNCS v2954*, pages 79–89. Springer.
- [Dwork et al., 2001] Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. (2001). Rank aggregation methods for the web. In *WWW 01: Proceedings of the 10th international conference on World Wide Web*, pages 613–622, New York, NY, USA. ACM.

- [Esuli and Sebastiani, 2010] Esuli, A. and Sebastiani, F. (2010). AI and opinion mining, part 2. *Intelligent Systems, IEEE*, 25(4):72–79.
- [Golub and Van Loan, 1996] Golub, G. H. and Van Loan, C. F. (1996). *Matrix Computations (Johns Hopkins Studies in Mathematical Sciences)*. The Johns Hopkins University Press.
- [Han et al., 2006] Han, J., Kamber, M., and Pei, J. (2006). *Data Mining: Concepts and Techniques, Second Edition (The Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann, 2 edition.
- [Hayes and Cunningham, 2004] Hayes, C. and Cunningham, P. (2004). Context boosting collaborative recommendations. In *Journal of Knowledge Based Systems, Volume 17, Issue*, pages 5–6. Elsevier.
- [Herlocker and Konstan, 2001] Herlocker, J. L. and Konstan, J. A. (2001). Content-independent task-focused recommendation. *IEEE Internet Computing*, 5(6):40–47.
- [Herlocker et al., 1999] Herlocker, J. L., Konstan, J. A., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *SIGIR*, pages 230–237. ACM.
- [Herlocker et al., 2000] Herlocker, J. L., Konstan, J. A., and Riedl, J. (2000). Explaining collaborative filtering recommendations. In *CSCW 00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250, New York, NY, USA. ACM.
- [Herlocker et al., 2004] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22:5–53.
- [Hill et al., 1995] Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In *CHI 95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.
- [Hitchcock, 1927] Hitchcock, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6.
- [Horozov et al., 2006] Horozov, T., Narasimhan, N., and Vasudevan, V. (2006). Using location for personalized POI recommendations in mobile environments. In *SAINT '06: Proceedings of the International Symposium on Applications on Internet*, pages 124–129, Washington, DC, USA. IEEE Computer Society.

- [Hu et al., 2008] Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *ICDM '08: Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, volume 0, pages 263–272, Washington, DC, USA. IEEE Computer Society.
- [Jameson and Smyth, 2007] Jameson, A. and Smyth, B. (2007). Recommendation to groups. In *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321, pages 596–627. Springer.
- [Jeong et al., 2009] Jeong, S., Kalasapur, S., Cheng, D., Song, H., and Cho, H. (2009). Clustering and naive bayesian approaches for situation-aware recommendation on mobile devices. In *International Conference on Machine Learning and Applications, ICMLA 2009, Miami Beach, Florida, USA, December 13-15, 2009*, pages 353–358.
- [Jin et al., 2004] Jin, R., Chai, J. Y., and Si, L. (2004). An automatic weighting scheme for collaborative filtering. In *SIGIR 04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 337–344, New York, NY, USA. ACM Press.
- [Joachims, 2002] Joachims, T. (2002). Optimizing search engines using click-through data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142.
- [Karatzoglou et al., 2010] Karatzoglou, A., Amatriain, X., Baltrunas, L., and Oliver, N. (2010). Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys '10: Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86, New York, NY, USA. ACM.
- [Karlson et al., 2010] Karlson, A. K., Iqbal, S. T., Meyers, B., Ramos, G., Lee, K., and Tang, J. C. (2010). Mobile taskflow in context: a screenshot study of smartphone usage. In *Proc. of CHI '10*.
- [Kohavi and John, 1997] Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324.
- [Koren, 2008] Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, KDD '08, pages 426–434, New York, NY, USA. ACM.
- [Koren, 2009] Koren, Y. (2009). Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, KDD '09, pages 447–456, New York, NY, USA. ACM.

- [Koren and Bell, 2010] Koren, Y. and Bell, R. (2010). Advances in collaborative filtering. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*. Springer.
- [Koren et al., 2009] Koren, Y., Bell, R. M., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8):30–37.
- [Lathauwer et al., 2000] Lathauwer, L. D., Moor, B. D., Vandewalle, J., and V, J. (2000). A multilinear singular value decomposition. *SIAM. J. Matrix Anal. & Appl.*, 21:1253–1278.
- [Lewis, 1995] Lewis, J. R. (1995). IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use. *International Journal of Human Computer Interaction*, 7(1):57–78.
- [Linden et al., 2003] Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76 – 80.
- [Lloyd, 1999] Lloyd, C. J. (1999). *Statistical Analysis Of Categorical Data*. Wiley-interscience.
- [Lops et al., 2010] Lops, P., de Gemmis, M., and Semeraro, G. (2010). Content-based recommender systems: State of the art and trends. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*. Springer.
- [Maes, 1994] Maes, P. (1994). Agents that reduce work and information overload. *Commun. ACM*, 37(7):30–40.
- [Manning, 2008] Manning, C. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge.
- [Marlin and Zemel, 2009] Marlin, B. M. and Zemel, R. S. (2009). Collaborative prediction and ranking with non-random missing data. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 5–12, New York, NY, USA. ACM.
- [Masthoff and Gatt, 2006] Masthoff, J. and Gatt, A. (2006). In pursuit of satisfaction and the prevention of embarrassment: affective state in group recommender systems. *User Modeling User-Adapted Interaction*, 16(3-4):281–319.
- [McCarthy, 2002] McCarthy, J. F. (2002). Pocket restaurantfinder: A situated recommender system for groups. In *In Proceedings of the Workshop on Mobile Ad-Hoc Communication at the 2002 ACM Conference on Human Factors in Computer Systems*. Minneapolis.

- [McCarthy and Anagnost, 1998] McCarthy, J. F. and Anagnost, T. D. (1998). Musicfx: an arbiter of group preferences for computer supported collaborative workouts. In *CSCW 98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 363–372, New York, NY, USA. ACM.
- [McCarthy et al., 2006] McCarthy, K., Salamó, M., Coyle, L., McGinty, L., Smyth, B., and Nixon, P. (2006). Cats: A synchronous approach to collaborative group recommendation. In *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference, Melbourne Beach, Florida, USA, May 11-13, 2006*, pages 86–91.
- [McKechnie, 1983] McKechnie, J. L., editor (December 1983). *Webster's New Twentieth Century Dictionary of the English Language*. Number 978-0671418199 in Dictionary. Simon and Schuster; 2nd edition.
- [McNee et al., 2006] McNee, S. M., Riedl, J., and Konstan, J. A. (2006). Being accurate is not enough: how accuracy metrics have hurt recommender systems. In Olson, G. M. and Jeffries, R., editors, *CHI Extended Abstracts*, pages 1097–1101. ACM.
- [Mcsherry, 2005] Mcsherry, D. (2005). Explanation in recommender systems. *Artificial Intelligence Review*, 24:179–197.
- [Mitchell, 1997] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.
- [MovieLens, 2007] MovieLens (2007). 100k movielens dataset, <http://www.grouplens.org/>. accessed in 2007.
- [O'Connor et al., 2001] O'Connor, M., Cosley, D., Konstan, J. A., and Riedl, J. (2001). Polylens: a recommender system for groups of users. In *ECSCW01: Proceedings of the Seventh European Conference on Computer Supported Cooperative Work*, pages 199–218, Norwell, MA, USA. Kluwer Academic Publishers.
- [Oku et al., 2006] Oku, K., Nakajima, S., Miyazaki, J., and Uemura, S. (2006). Context-aware svm for context-dependent information recommendation. In *Proceedings of the 7th international Conference on Mobile Data Management*.
- [Oku et al., 2007] Oku, K., Nakajima, S., Miyazaki, J., and Uemura, S. (2007). Investigation for designing of context-aware recommendation system using svm. In Ao, S. I., Castillo, O., Douglas, C., Feng, D. D., and Lee, J.-A., editors, *IMECS*, Lecture Notes in Engineering and Computer Science, pages 970–975. Newswood Limited.
- [Ono et al., 2009] Ono, C., Takishima, Y., Motomura, Y., and Asoh, H. (2009). Context-aware preference model based on a study of difference between real and

- supposed situation data. In *UMAP 09: Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization*, pages 102–113, Berlin, Heidelberg. Springer-Verlag.
- [Panniello et al., 2009] Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C., and Pedone, A. (2009). Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM Conference on Recommender Systems*, RecSys '09, pages 265–268, New York, NY, USA. ACM.
- [Pazzani and Billsus, 2007] Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321, pages 325–341. Springer.
- [Peddy and Armentrout, 2003] Peddy, C. C. and Armentrout, D. (2003). *Building Solutions with Microsoft Commerce Server 2002*. Microsoft Press, Redmond, WA, USA.
- [Peer, 2010] Peer, S. (2010). Real-time context-aware recommendations for mobile users. Master's thesis, Free University of Bolzano.
- [Pizzutilo et al., 2005] Pizzutilo, S., De Carolis, B., Cozzolongo, G., and Ambrosio, F. (2005). Group modeling in a public space: methods, techniques, experiences. In *AIC05: Proceedings of the 5th WSEAS International Conference on Applied Informatics and Communications*, pages 175–180, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS).
- [Pu et al., 2008] Pu, P., Bridge, D. G., Mobasher, B., and Ricci, F., editors (2008). *Proceedings of the 2008 ACM Conference on Recommender Systems, RecSys 2008, Lausanne, Switzerland, October 23-25, 2008*. ACM.
- [Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning (Morgan Kaufmann Series in Machine Learning)*. Morgan Kaufmann, 1 edition.
- [Rendle et al., 2009] Rendle, S., Balby Marinho, L., Nanopoulos, A., and Schmidt-Thieme, L. (2009). Learning optimal ranking with tensor factorization for tag recommendation. In *KDD 09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736, New York, NY, USA. ACM.
- [Rennie and Srebro, 2005] Rennie, J. D. M. and Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *In Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 713–719. ACM.

- [Resnick et al., 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer Supported Cooperative Work*, pages 175–186, New York, NY, USA. ACM.
- [Resnick and Varian, 1997] Resnick, P. and Varian, H. R. (1997). Recommender systems. *Commun. ACM*, 40(3):56–58.
- [Ricci, 2010] Ricci, F. (2010). Mobile recommender systems. *JITT*.
- [Ricci et al., 2010] Ricci, F., Rokach, L., and Shapira, B. (2010). Introduction to recommender systems handbook. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P., editors, *Recommender Systems Handbook*, pages 1–35. Springer Verlag.
- [Rubens et al., 2010] Rubens, N., Kaplan, D., and Sugiyama, M. (2010). Active learning in recommender systems. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P., editors, *Recommender Systems Handbook*. Springer Verlag. in press.
- [Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA. ACM.
- [Sarwar et al., 2000] Sarwar, B. M., Karypis, G., Konstan, J. A., and Riedl, J. T. (2000). Application of dimensionality reduction in recommender systems—a case study. In *ACM WebKDD Workshop*.
- [SAS, 1999] SAS (1999). SAS Institute Inc., SAS OnlineDoc(TM), Version 7-1 Cary, NC: SAS Institute Inc., 1999.
- [Schafer et al., 2007] Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative filtering recommender systems. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, chapter 9, pages 291–324. Springer, Berlin.
- [Schafer et al., 1999] Schafer, J. B., Konstan, J., and Riedl, J. (1999). Recommender systems in e-commerce. In *EC '99: Proceedings of the 1st ACM conference on Electronic Commerce*, pages 158–166, New York, NY, USA. ACM.
- [Schlar et al., 2009] Schlar, A., Tsikinovsky, A., Rokach, L., Meisels, A., and Antwarg, L. (2009). Ensemble methods for improving the performance of neighborhood-based collaborative filtering. In *RecSys '09: Proceedings of the*

- third ACM conference on Recommender Systems*, pages 261–264, New York, NY, USA. ACM.
- [Schein et al., 2002] Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, New York, NY, USA. ACM.
- [Schilit et al., 1994] Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. *Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on*, pages 85–90.
- [Schilit and Theimer, 1994] Schilit, B. and Theimer, M. (1994). Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32.
- [Shani and Gunawardana, 2010] Shani, G. and Gunawardana, A. (2010). Evaluating recommendation systems. In Ricci, F., Rokach, L., and Shapira, B., editors, *Recommender Systems Handbook*, pages 257–298. Springer Verlag.
- [Shardanand and Maes, 1995] Shardanand, U. and Maes, P. (1995). Social information filtering: Algorithms for automating “word of mouth”. In *Proceedings of ACM CHI 95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217.
- [Shi et al., 2010] Shi, Y., Larson, M., and Hanjalic, A. (2010). Mining mood-specific movie similarity with matrix factorization for context-aware recommendation. In *Challenge on Context-aware Movie Recommendation*.
- [Srebro et al., 2005] Srebro, N., Rennie, J., and Jaakkola, T. (2005). Maximum-margin matrix factorization. In Saul, L. K., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, Cambridge, MA. MIT Press.
- [Srebro and Shraibman, 2005] Srebro, N. and Shraibman, A. (2005). Rank, trace-norm and max-norm. In Auer, P. and Meir, R., editors, *Proc. Annual Conf. Computational Learning Theory*, number 3559 in Lecture Notes in Artificial Intelligence, pages 545–560. Springer-Verlag.
- [Sungrim Kim, 2007] Sungrim Kim, J. K. (2007). Effective context-aware recommendation on the semantic web. *IJCSNS - International Journal of Computer Science and Network Security*, 7(8):154–159.
- [svdlib, 2008] svdlib (2008). Doug Rohde’s SVD C Library, version 1.34, <http://tedlab.mit.edu/~dr/svdlbc/>. Accessed in 2008.

- [Swarbrooke and Horner, 2007] Swarbrooke, J. and Horner, S. (2007). *Consumer Behaviour in Tourism*. Elsevier Ltd.
- [Timeley, 2008] Timeley (2008). Timely Development implementation, <http://www.timelydevelopment.com>. Retrieved in year 2008.
- [Tintarev and Masthoff, 2010] Tintarev, N. and Masthoff, J. (2010). Designing and evaluating explanations for recommender systems. In Ricci, F., Rokach, L., Shapira, B., and Kantor, P. B., editors, *Recommender Systems Handbook*. Springer.
- [Tversky and Kahneman, 1991] Tversky, A. and Kahneman, D. (1991). Loss Aversion in Riskless Choice: A Reference-Dependent Model. *The Quarterly Journal of Economics*, 106(4):1039–1061.
- [van Setten et al., 2004] van Setten, M., Pokraev, S., and Koolwaaij, J. (2004). Context-aware recommendations in the mobile tourist application compass. In Bra, P. D. and Nejdl, W., editors, *AH*, volume 3137 of *Lecture Notes in Computer Science*, pages 235–244. Springer.
- [Weimer et al., 2007] Weimer, M., Karatzoglou, A., Le, Q. V., and Smola, A. J. (2007). Cofi rank - maximum margin matrix factorization for collaborative ranking. In Platt, J. C., Koller, D., Singer, Y., and Roweis, S. T., editors, *NIPS*. MIT Press.
- [Weimer et al., 2008] Weimer, M., Karatzoglou, A., and Smola, A. (2008). Improving maximum margin matrix factorization. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, ECML PKDD '08, pages 263–276, Berlin, Heidelberg. Springer-Verlag.
- [Xiong et al., 2010] Xiong, L., Chen, X., Huang, T., and J. Schneider, J. G. C. (2010). Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of SIAM Data Mining*.
- [Yahoo, 2007] Yahoo (2007). Yahoo! Research Webscope Movie Data Set. <http://research.yahoo.com/>, Version 1.0. Received in 2007.
- [Yao, 1995] Yao, Y. (1995). Measuring retrieval effectiveness based on user preference of documents. *Journal of the American Society for Information Science*, 46(2):133–145.
- [Yap and Pang, 2007] Yap, G.-E. and Pang, H.-H. (2007). Discovering and exploiting causal dependencies for robust mobile context-aware recommenders. *IEEE Trans. on Knowl. and Data Eng.*, 19(7):977–992. Senior Member-Ah-Hwee Tan.

- [Yap et al., 2005] Yap, G.-E., Tan, A.-H., and Pang, H.-H. (2005). Dynamically-optimized context in recommender systems. In *MDM 05: Proceedings of the 6th International Conference on Mobile Data Management*, pages 265–272, New York, NY, USA. ACM.
- [Young and Levenglick, 1978] Young, H. P. and Levenglick, A. (1978). A consistent extension of Condorcet’s election principle. *SIAM Journal on Applied Mathematics*, 35(2):285–300.
- [Yu et al., 2003] Yu, K., Xu, X., Ester, M., and Kriegel, H.-P. (2003). Feature weighting and instance selection for collaborative filtering: An information-theoretic approach*. *Knowl. Inf. Syst.*, 5(2):201–224.
- [Yu et al., 2006a] Yu, Z., Zhou, X., Hao, Y., and Gu, J. (2006a). Tv program recommendation for multiple viewers based on user profile merging. *User Modeling and User-Adapted Interaction*, 16(1):63–82.
- [Yu et al., 2006b] Yu, Z., Zhou, X., Zhang, D., Chin, C.Y. and Wang, X., and Men, J. (2006b). Supporting context-aware media recommendations for smart phones. *IEEE Pervasive Computing*, 5(3):68–75.