Exploiting Distributional Semantics for Content-Based and Context-Aware Recommendation

PhD Thesis in Artificial Intelligence

Victor Codina

Advisor: Dr. Luigi Ceccaroni

Barcelona, May 2014

Universitat Politècnica de Catalunya - BarcelonaTech
Departament de Llenguatges i Sistemes Informàtics,

Contents

List of Figures

List of Tables

Abstract

During the last decade, the use of recommender systems has been increasingly growing to the point that, nowadays, the success of many well-known services depends on these technologies. Recommenders Systems help people to tackle the choice overload problem by effectively presenting new content adapted to the user's preferences. However, current recommendation algorithms commonly suffer from data sparsity, which refers to the incapability of producing acceptable recommendations until a minimum amount of users' ratings are available for training the prediction models.

This thesis investigates how the distributional semantics of concepts describing the entities of the recommendation space can be exploited to mitigate the *data-sparsity* problem and improve the prediction accuracy with respect to state-of-the-art recommendation techniques. The fundamental idea behind distributional semantics is that concepts repeatedly co-occurring in the same context or usage tend to be related. In this thesis, we propose and evaluate two novel semantically-enhanced prediction models that address the sparsity-related limitations: (1) a content-based approach, which exploits the distributional semantics of item's attributes during item and user-profile matching, and (2) a context-aware recommendation approach that exploits the distributional semantics of contextual conditions during context modeling. We demonstrate in an exhaustive experimental evaluation that the proposed algorithms outperform state-of-the-art ones, especially when data are sparse.

Finally, this thesis presents a recommendation framework, which extends the widespread machine learning library Apache Mahout, including all the proposed and evaluated recommendation algorithms as well as a tool for offline evaluation and meta-parameter optimization. The framework has been developed to allow other researchers to reproduce the described evaluation experiments and make new progress on the Recommender Systems field easier.

Acknowledgements

First of all, I would like to express my sincere gratitude to my advisor, Dr. Luigi Ceccaroni, for the continuous support and mentoring during all these years. Without his guidance and persistent help this PhD thesis would not have been possible. It has been a big pleasure and benefit to work with such a strong and great-personality researcher, helping me to develop the essential skills for research.

I am deeply grateful to Prof. Francesco Ricci for offering me the opportunity to collaborate with him during the last two years of my PhD, and accepting my request to do an internship in his research group at the Free University of Bozen-Bolzano. Since the first time we met, he has been extraordinarily supportive and generous. His constructive feedback and stimulating discussions have been an enormous help to improve the quality of this research and expand my knowledge in the field of Recommender Systems. Special thanks also to the members of the DIS research group with whom I shared unforgettable experiences during my stay at Bolzano.

I also want to thank my colleagues and members of the KEMLG research group for the good time we spent together. I am particularly grateful to Prof. Javier Vázquez for his guidance and support, especially for helping me to carry out my internship at the Free University of Bozen-Bolzano. In addition, I would like to acknowledge the financial, academic and technical support provided by the Universitat Politècnica de Catalunya - BarcelonaTech, particularly in the award of a pre-graduate research grant (FPI-UPC) which provided the necessary financial support for this research.

Last but not the least, I would like to express my deepest gratitude to my friends, for always being there, to my wife Gemma, for her personal support and great patience at all times, and to my parents, René and Joana, for their continuous spiritual support and trust in me. I dedicate this work to you.

Chapter 1 - Introduction

1.1. Background

The overwhelming number of alternative items (e.g. movies, books, music, food, and news) available nowadays on many services makes harder the users' decision-making process. In order to tackle this *choice overload* problem, several software tools have been proposed that help users by presenting content adapted to their particular interests and needs.

Information Retrieval (IR) systems were the first technology solutions to the problem of choice overload [Manning et al. 2009]. The goal of these systems is to perform an efficient retrieval upon a huge quantity of items taking into account a given query or need of a user. Items are commonly annotated with keywords describing some aspects of their content (e.g. for web documents, the most representative terms may be used as annotations). Search engines (e.g. Google, Yahoo!, and Bing) are the most common and maybe best instantiation of IR systems. These systems present a ranked list of potentially relevant items based on the similarities between the user query and the item annotations as well as other relevancy measures. However, a main limitation of these search-based systems is their assumption that users are always aware of their information needs and know how to express them with queries. Many times people discover facts of whose existence were not aware, but which are very interesting for them. As Jeffrey M, O'Brien said in his article "The race to create a 'smart' Google" published in CNN Money on November 2006: "*We are leaving the era of search and entering one of discovery*. *What is the difference? Search is what you do when you are looking for something. Discovery is when something wonderful that you did not know existed, or did not know how to ask for, finds you"*.

In the mid-nineties, Recommender Systems (RSs) emerge as an independent research field of IR and Artificial Intelligence (AI), to address the *choice overload* problem through recommendations instead of information searching [Ricci et al. 2011]. The development of RSs was motivated by the fact that most users naturally tend to solve the choice overload problem by relying on other's recommendations. For example, when selecting a movie to watch, users commonly rely on the reviews of film critics or other trustworthy users. Therefore, given the importance of recommendations on the users' decision-making process, the main goal of RSs is to improve the quality of recommendations by adapting them to the users' interests. To accomplish this task, RSs estimate the relevance of those items that are yet to be consumed by users taking into account the feedback provided during their previous interactions with the system. During the last decade, the use of these systems has been increasingly growing to the point that, nowadays, the success of many well-known services strongly depends on these technologies. **Table 1.1** shows some examples of RSs in different application domains.

Table 1.1. Examples of recommender systems.

Application domain	Recommender System	Website
Books	Amazon	http://www.amazon.com
	LibraryThing	http://www.librarything.com
Movies	Netflix	http://www.netflix.com
	MovieLens	http://www.movielens.org
IPTV	TiVo	http://www.tivo.com
Travel	Google Now	http://www.google.com/landing/now/
Songs	Pandora	http://www.pandora.com
	Last fm	http://www.last.fm
News	Google News	https://news.google.com
	Yahoo! News	http://news.yahoo.com

In the literature, several recommendation techniques[1] have been proposed, each of which has its own advantages and drawbacks [Adomavicius and Tuzhilin 2005]. Two main families of recommender systems are commonly distinguished: Content-Based filtering (CB), which expresses user's tastes as attribute-based user profiles and recommend new items matching user and item attribute-based profiles; and Collaborative Filtering (CF), which bases predictions on the feedback given by other users with similar rating behavior to the target one. In practice, real systems like the ones presented in **Table 1.1,** employ a combination of techniques (i.e. hybrid strategies) trying to enforce their advantages and mitigate their limitations.

1.2. Research Motivation

A limitation of most recommendation techniques is their lack of *context-awareness*, which means that they fail to adapt the recommendations to the users' context. Context is a multifaceted concept that has been studied across different research disciplines, including AI, ubiquitous computing, cognitive science, linguistics, philosophy, psychology, and organizational sciences. Since we focus on the use of context in recommender systems, in this thesis we adopt the definition proposed by Dey [2001]: "*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*".

In most application domains contextual information can play an important role in order to produce more accurate and "intelligent" recommendations. For instance, when recommending a beach or a natural park the current weather may cause a completely different user's experience (e.g. liking the recommendation when sunny but detesting it when snowing). The same can happen when recommending

[1] Thorough this document, the terms "recommendation techniques", "recommendation algorithms", "recommendation models" and "prediction models" are used as synonyms.

movies, books and songs. Users can have different movie, music and book preferences depending on their context, such as if they are working or on holidays, if they are alone or with some friends, and the period of the year.

Context-Aware Recommender Systems (CARSs) belong to an active and growing research area, which aims at improving traditional (context-free) recommenders (both CB or CF techniques) by exploiting the contextual information under which the users' experienced or rated the items. In the last years, several CARS techniques have been proposed that outperform context-free approaches when they incorporate relevant contextual information into their prediction models [Adomavicius and Tuzhilin 2011; Adomavicius et al. 2011]. Recent empirical evaluations indicate that no context-aware approach is universally dominant, and the best performing method depends on the recommendation task and domain [Panniello et al. 2014]. Their analysis also shows that the accuracy of current CARS techniques decreases when contextual information has a finer granularity (i.e. it is more specific) and thus fewer ratings acquired in the target contextual situation are available. This *data-sparsity* limitation[2] affects especially to CARS techniques because they require for a large training set of ratings of users for items in different contextual situations in order to learn robust context-aware prediction models.

In this thesis, we claim that it is possible to address such a limitation of current CARS techniques by exploiting semantic similarities between contextual situations when making recommendations in a target situation. For instance, if we want to predict the rating for a Place of Interest (POI), e.g., the "Sagrada Familia" church (in Barcelona, Spain), and the target contextual situation (i.e. the situation for which the system has to make the recommendation) includes a condition such as, "group composition" is "two adults and two children", ratings acquired when the "group composition" was "two adults and three children" may also be used to generate an accurate predictive model in the target context. Most current techniques do not exploit this idea of reusing ratings acquired in "syntactically" different situations because they employ syntactic-based matching techniques, which implies that their domain-specific semantic similarities are not taken into account during contextual modeling.

The need for semantics understanding and exploitation in recommender systems to produce better recommendations has been manifested in several works in the last decade [Mobasher et al. 2003; Middleton et al. 2004; Cantador et al. 2008; Sieg et al. 2010]. The majority of these semantically-enhanced recommender systems (SERSs) employ ontologies to represent the semantic knowledge because it allows for a simple, scalable and portable description of the concepts belonging to the recommendation domain and their relationships. Mostly, SERSs have focused on improving the effectiveness of traditional CB techniques, based on "syntactic" keyword matching methods, by

[2] Some researchers also refer to this limitation as a type of *cold-start* scenario in a wider sense [Schein et al. 2002]. In a strict sense, *cold-start* scenarios are those situations where either the items or the users have no rating associated at all.

exploiting the explicit semantic relationships between the contents of the items to recommend. However, the expressiveness and richness of available ontologies are often limited given the difficulty of manually modeling all the existing semantic relationships between concepts of particular domain at design stage. In addition, most of the available ontologies basically consist of taxonomies that classify the items of a system into several categories, since this is the natural way of classify content on the large e-commerce websites, and this limits to a great extent the improvement that can be achieved by using ontology-based SERS techniques.

An alternative to ontology-based semantics is data-driven distributional semantics, where the "meaning" of a concept is represented based on its distributional properties that can be automatically derived from the data. The fundamental idea behind this use of data to extract semantic similarities between domain concepts is the so-called **distributional hypothesis:** *concepts repeatedly co-occurring in the same context or usage tend to be related*.

In this thesis, we claim that, when ontologies available for recommendation consist of general domain taxonomies, which do not express deep domain-specific relations between concepts, such as the movie genre classifications available on Amazon and Netflix sites, similarities based on distributional semantics are more useful than ontology-based ones to enhance the accuracy of recommender systems. The following is an example of such a context taxonomy in the movie domain, defining the hierarchical relationships of *time* and *company* conditions when having a movie-watching experience [Adomavicius et al. 2005]:

- Time: "Saturday" → "Weekend" → "Any time"
- Company: "Boyfriend" → "Friends" → "Not alone" → "Any company"

To illustrate the idea of distributional semantics we return to the example of the context-aware tourism recommender system. Let's suppose that now the target contextual situation contains the *sunny* day condition. Should the system also consider ratings provided to POIs when travelling with *family* in that situation? Clearly, it is difficult to find out such specific condition-to-condition associations based on general context taxonomies like the one mentioned above. In this case, the easier way to obtain such similarities is by directly looking at the rating data. Let's assume that we analyze how *sunny* and *family* conditions influence the users' ratings and we discover the following influencing pattern: both conditions tend to increase the user's ratings for indoor places like museums, and decrease the ratings for outdoor places like castles. Based on this similar influence pattern of both conditions with respect to the same type of items (i.e. similar distributional semantics), we may confidently assess that these two conditions are highly related in this particular application domain.

Analogously, we also claim that distributional semantics of items' attributes can be exploited for improving more effectively than ontology-based SERS techniques the prediction accuracy of context-free

CB techniques. This intuition relies on the same premise as for context-aware recommendation: similarities based on distributional semantics are data-dependent, and thus suit better the rating data that are actually used by the system for making recommendations. Differently from semantics of contextual conditions, in this case the distributional properties of items' attributes cannot be based on how they influence the user's ratings. Instead here we are more interested in analyzing the "usage" pattern of the attributes across the system's users and items. For example, let's imagine that the profile of the target user only contains a positive interest in *Bruce Willis* movie actor, and the movie to recommend is an *action* movie. Based on these attribute-based profiles, would the CB recommender suggest this movie to the user? Using a traditional CB technique based on "syntactic" keyword-based matching the answer would be "no", and it is likely that a SERS technique exploiting ontology-based semantics also does not have any evidence to recommend it, unless some actors and genres are explicitly related in the ontology. Now, let's assume that we know that several users of the system that like *action* movies also like *Bruce Willis* in a similar way. In this case the system could infer that a strong association exists between these two attributes; therefore, it would probably recommend the movie to the user even with only this information available, alleviating thus the data-sparsity problem.

1.3. Goals

Taking into account the main limitations of current techniques in addressing the data-sparsity problem, presented in the previous section, the ultimate goal of this thesis is the following:

To develop novel recommendation algorithms capable of exploiting distributional similarities between concepts describing items or contextual situations to overcome the sparsity-related limitations of state-of-the-art approaches and thus produce more accurate recommendations.

To fulfill this general goal, this thesis has the following two research objectives:

1. **The development of content-based recommendation algorithms enhanced with distributional semantics of items' attributes that can outperform the state-of-the-art CB and SERS techniques**. We shall investigate novel methods to acquire and exploit attribute-to-attribute similarities based on their distributional semantics in the CB recommendation process. We will analyze whether the common methods used for exploiting ontology-based semantics are also appropriate when used with distributional semantics. Finally, we shall assess the improvements and benefits of the proposed methods based on distributional semantics compared to the ontology-based ones and other state-of-the-art SERS techniques. We aim to perform this comparison appropriately and in a reproducible way by using publicly available data sets in the most common recommendation domains.

2. **The development of context-aware recommendation algorithms enhanced with distributional semantics of contextual conditions that can outperform the state-of-the-art CARS techniques**. We shall investigate novel methods to acquire and exploit condition-to-condition similarities based on their distributional semantics during context modeling. State-of-the-art CARS techniques typically incorporate contextual information directly into Matrix Factorization (MF) prediction models, a specific CF approach whose popularity has increased in the last years because of its superior accuracy and scalability. Therefore, to outperform the state of the art we shall study how the distributional semantics of contextual conditions can be exploited on top of MF models.

1.4. Summary of Contributions

This section describes the main contributions of this thesis to the Recommender Systems field:

1. A novel SERS technique, *Semantic Content-Based* (SCB) filtering, which mitigates the sparsity-related limitations of current CB recommendation algorithms and outperforms state-of-the-art SERS techniques by exploiting the distributional similarities between item's attributes during the key process of user and item profile matching.
2. A novel SERS approach to context-aware recommendation, *Semantic Pre-Filtering* (SPF), which is able to tackle the data-sparsity problem and outperform state-of-the-art CARS techniques by exploiting distributional semantic similarities between contextual situations during context modeling.
3. The *extension of a free open-source recommendation framework* including all the implemented and evaluated recommendation algorithms as well as a tool for offline evaluation of the available algorithms, allowing other researchers to reproduce the empirical evaluation carried out in this thesis and make new progress on the RS field easier.

1.5. Publications Related to this Thesis

Here we present the list of publications related to this thesis grouped by type of publication and ordered by date:

Conference papers:

1. Codina, V., & Ceccaroni, L. (2010a). A Recommendation System for the Semantic Web. In A. Ponce de Leon F. de Carvalho, S. Rodríguez-González, J. F. De Paz, & J. . Corchado (Eds.), *Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence* (pp. 45–52). September 7-10, 2010, Valencia, Spain: Springer.
2. Codina, V., & Ceccaroni, L. (2010b). Taking advantage of semantics in recommendation systems. In R. Alquézar, A. Moreno, & J. Aguilar (Eds.), *Proceedings of the International*

Congress of the Catalan Association of Artificial Intelligence (pp. 163 – 172). October 20-22, 2010, L'Espluga de Francolí, Spain: IOS Press, Amsterdam, The Netherlands.

3. Codina, V., & Ceccaroni, L. (2011). Extending Recommendation Systems with Semantics and Context-Awareness : Pre-Filtering Algorithms. In C. Fernández, H. Geffner, & F. Manyà (Eds.), *Proceedings of the International Congress of the Catalan Association of Artificial Intelligence* (pp. 81–90). October 26-28, 2011, Lleida, Spain: IOS Press, Amsterdam, The Netherlands.
4. Codina, V., & Ceccaroni, L. (2012). Semantically-Enhanced Recommenders. In D. Riaño, E. Onaindia, & M. Cazorlam (Eds.), *Proceedings of the International Congress of the Catalan Association of Artificial Intelligence* (pp. 69–78). October 24-26, 2012, Alicante, Spain: IOS Press, Amsterdam, The Netherlands.
5. Codina, V., Ricci, F., & Ceccaroni, L. (2013a). Exploiting the Semantic Similarity of Contextual Situations for Pre-filtering Recommendation. In S. Carberry, S. Weibelzahl, A. Micarelli, & G. Semeraro (Eds.), *Proceedings of the 21th International Conference on User Modeling, Adaptation, and Personalization (UMAP'13)* (pp. 165–177). June 10-14, Rome, Italy: Springer, Berlin Heidelberg.
6. Codina, V., Ricci, F., & Ceccaroni, L. (2013b). Local Context Modeling with Semantic Pre-filtering. In *Proceedings of the 7th ACM conference on Recommender systems (RecSys'13)* (pp. 363–366). October 14-16, 2013, Hong Kong: ACM New York, NY, USA.
7. Codina, V., Ricci, F., & Ceccaroni, L. (2013c). Semantically-enhanced pre-filtering for context-aware recommender systems. In *Proceedings of the 3rd Workshop on Context-awareness in Retrieval and Recommendation* (pp. 15–18). February 5, Rome, Italy: ACM New York, NY, USA.

Journal article:

8. Codina, V., Ricci, F., & Ceccaroni, L.. Using Semantic Pre-filtering for Context-Aware Recommendation: an Experimental Evaluation. *In User Modeling and User-Adapted Interaction (UMUAI)* (current status: accepted).

1.6. Organization of the Thesis

In addition to this introductory chapter, this thesis is organized as follows:

- **Chapter 2 -Recommender Systems-** presents the state-of-the-art recommendation algorithms that do not incorporate semantic knowledge into their processes. We describe the major approaches to content-based, collaborative filtering, context-aware and hybrid recommendation. In the following chapters, some of the recommendation algorithms described in this chapter are used for performance comparisons.
- **Chapter 3 -Semantically-Enhanced Recommender Systems-** describes the related work that is more specific to this thesis, describing the state-of-the-art SERS approaches to incorporate semantic knowledge, mostly based on ontologies, into existing CB recommendation algorithms and CARS techniques in order to mitigate their lack of semantics understanding and exploitation, and thus improve their performance**.** This chapter also presents the empirical evaluation of an

existing ontology-based SERS technique that we carried out to better understand the limitations of the state of the art. The results of this experimentation were presented in Codina and Ceccaroni [2010a; 2010b].

- **Chapter 4 -Exploiting Distributional Similarities for Enhanced Content-Based Recommendation-** presents SCB, the proposed SERS technique to CB recommendation that mitigates the *data-sparsity* limitation by exploiting the distributional semantics of items' attributes during profile matching (in the prediction phase). We also present an offline evaluation of SCB using a well-known movie-rating data set, comparing their performance to state-of-the-art SERS, and also analyzing the effectiveness of distributional semantics compared to ontology-based semantics for improving the accuracy of CB recommendation. An initial definition of this technique was presented in Codina and Ceccaroni [2011], and a more complete description and evaluation of SCB was published in Codina and Ceccaroni [2012].
- **Chapter 5 -Exploiting Distributional Semantics for enhanced Context-Aware Recommendation-** presents SPF, the proposed SERS technique to context-aware recommendation that alleviates the data-sparsity limitation by exploiting the distributional semantics of contextual conditions during context modeling. We present an exhaustive offline evaluation of SPF using several contextually-tagged data sets, comparing its prediction accuracy to state-of-the-art CARS. Different variants of SPF were introduced by Codina et al. [2013a; 2013b; 2013c], and an extended version of the results of this research was submitted to the *User Modeling and User-Adapted Interaction* journal [Codina et al. submitted], which is currently under review.
- **Chapter 6 -Developed Recommendation Framework-** describes the additional functionality we have included to an existing recommendation framework for supporting the development and evaluation of the state-of-the-art CB algorithms, as well as CARS and SERS techniques (including the ones proposed in this thesis). This chapter also illustrates how to use the provided recommendation framework and tool for offline evaluation.
- **Chapter 7 -Conclusions-** closes the thesis presenting the main contributions as well as future research lines to be investigated.

Appendixes have been included, containing additional information:

- **Appendix A -Acronyms-** lists the meaning of all the acronyms used in this document.
- **Appendix B -Description of Contextual Information-** provides a more detailed description of the contextual information represented in some of the contextually-tagged data sets used for evaluating SPF, but it is not crucial for understanding the main content of the thesis.

Chapter 2 - Recommender Systems

2.1. Introduction

Recommender Systems (RSs) are information processing systems that actively gather historical data of user's behavior in order to recommend items of interest to the users. This chapter presents an in-depth description of the major recommendation techniques in the RSs field, which we classified into five main families: baseline predictors (Section 2.2), Content-Based Filtering (CB) (Section 2.3), Collaborative Filtering (CF) (Section 2.4), Context-Aware Recommendation Systems (CARSs) (Section 2.5) and Hybrid strategies (Section 2.6)

The traditional recommendation problem is commonly formulated as the problem of estimating ratings for the items that have not been consumed/rated/seen by a user [Adomavicius and Tuzhilin 2005; Ricci et al. 2011]. To accomplish this task, the recommender usually learns the users' preferences from their previous ratings to other items as well as additional information about the users (e.g. user's demographics) and/or the items (e.g. item's attributes). More formally, given a set of users U and a set of possible items to recommend I, and given incomplete rating data $r: U \times I \rightarrow R$, the recommendation algorithm consists of an estimation function $\hat{r}: U \times I \rightarrow R$ which estimates the rating of a given user for a given item. Normally, predicted ratings ($\hat{r}_{ui}$) are values in the same range as the actual user ratings (e.g. using a 5-star rating scale R = [1, 5]). However, if the output of the recommendation is a ranked list of items, then it is not necessary that the predicted value is limited to a specific range because in this case what matters most is the order in which the items are presented to the user.

In the RS field, rating is a general term that is used to represent a positive or negative user-item interaction, which can be explicitly or implicitly collected. Explicit ratings can take a variety of forms: numerical ratings like the well-known 1-5 stars, ordinal ratings such as "agree, neutral, disagree", binary ratings in which the user is asked to decide if a certain item is good or bad, and unary ratings like the Facebook's thumb up icon that it is used to explicitly show a positive interest in a specific item. Implicit ratings commonly take the unary form and basically consist of user's actions, such as searching for more information about an item or buying an item.

2.2. Baseline Predictors

Baseline predictors (also known as non-personalized recommenders) are usually naïve recommendation techniques that make recommendations without considering the user-item interactions. A well-known example is the *most-popular* technique where items are simply ranked according to their popularity across all the users of the system.

Baseline predictors are especially useful when the recommendation task consists of predicting user's ratings because they capture those effects associated to users and items that cannot be inferred from the user-item interactions. For this reason, baseline predictors are commonly used in combination with well-known CB or CF recommendation algorithms in order to produce more accurate predictions [Koren and Bell 2011].

The most commonly used baseline predictor is the one that only models the systematic tendencies or biases associated to the users when giving ratings (e.g. a critic user that rates lower than others) and some items when receiving ratings (e.g. a popular items that receives higher ratings than usual). Denoting by μ the overall rating average, this baseline predictor models the bias associated to a certain user *u* and item *i* as follows:

$$b_{ui} = \mu + b_u + b_i \tag{2.1}$$

The parameters b_u and b_i are the observed deviations of user *u* and item *i*, respectively, from the average. For example, let's suppose that we want to predict the baseline rating of the movie *Matrix* by user *John*, and that the average rating over all movies (μ) is 3.5 stars. Assuming that based on the available ratings, the system knows that *Matrix* is a popular movie that tends to be rated 0.5 stars above the average (b_i), and *John* is a critical user who tends to rate 0.3 stars lower than the average (b_u), the baseline prediction in this case would be 3.7 stars.

Basically, two different methods can be used in order to learn these parameters for all users and items. One method consists of simply using average offsets, computing subsequent effects within the residuals of previous effects as follows:

$$b_u = \frac{1}{|I_u| + \beta} \sum_{i \in I_u} (r_{ui} - \mu) \tag{2.2}$$

$$b_i = \frac{1}{|U_i| + \beta} \sum_{u \in U_i} (r_{ui} - b_u - \mu) \tag{2.3}$$

Here the damping term β is used to provide a more reasonable estimate for the users and items that have few ratings associated (i.e. cold-start users or items), since this parameter makes the predicted value to be closer to 0 and hence closer to the global average μ in those cases. β is based on the following statistical principle: the more ratings a user has provided or an item has received, the more trustworthy is the predicted user's or item's bias. β is a data-dependent meta-parameter which has to be experimentally identified.

A more sophisticated method to learn the parameters is by solving the following least squares optimization problem:

$$\min_{b*} \sum_{r_{ui} \in R} \left[(r_{ui} - \mu - b_u - b_i)^2 + \lambda \left(b_u^2 + b_i^2 \right) \right] \tag{2.4}$$

This function is typically used to optimize the model parameters for rating prediction. The minimization is normally performed by using stochastic gradient descent (SGD), a method which was popularized by Funk [2006] during the Netflix prize challenge. Basically, the SGD method consists of randomly iterating over all the ratings in the training set, and for each rating r_{ui} this three-step process is repeated:

(1) a prediction ($\hat{r}_{ui}$) is produced with the current model parameters;

(2) the associated prediction error $e_{ui} = r_{ui} - \hat{r}_{ui}$ is computed; and

(3) in order to reduce the rating prediction error, each model parameter is modified in the opposite direction of the gradient (i.e. partial derivatives) of the objective function. In the case of the baseline predictor the gradient is simply the associated predictor error e_{ui}, and the user and item biases are modified as follows:

$$b_u \leftarrow b_u + \gamma(e_{ui} - \lambda \cdot b_u) \tag{2.5}$$

$$b_i \leftarrow b_i + \gamma(e_{ui} - \lambda \cdot b_i) \tag{2.6}$$

The constant λ controls the extent of regularization that is used to prevent over-fitting, and γ is the *learning rate*, i.e. determines the size of the steps when moving in the opposite direction of the gradient. These meta-parameters of the SGD learning process must be tuned experimentally and it is advised to use distinct values for each type of parameter (e.g. one for b_u and b_i). In the case of the learning rate, it is also a good practice to employ a decay strategy to reduce its value after each iteration/epoch, since it helps to converge easily. As we will see in next sections, baseline predictors are usually incorporated into well-known CF or CB prediction models.

2.3. Content-Based Filtering

Content-based recommendation algorithms rely on the assumption that users may like items whose attributes are similar to those of the other items which users have previously rated positively [Pazzani and Billsus 2007; Lops et al. 2011]. This approach has a straightforward justification: recommend items to a user if she is interested in some attributes of the item. For example, if we know that *Matrix* is a *science fiction* action movie, and *John* loves this movie genre, it would be intuitive to recommend this movie to him.

Inspired by IR, the problem of CB recommendation has been traditionally considered as a specific type of text classification problem, where the goal is to classify an unseen document as relevant or non-relevant (user likes/dislikes) based on the previous users' ratings and the document descriptions. For this reason, most CB recommendation techniques are adaptations of text classification algorithms.

The general process of CB recommendation can be divided into two main components: (1) *content analysis* (Section 2.3.1), in which raw item descriptions (i.e. unstructured item data) are analyzed and represented in a structured way (vector space); and (2) *user-profile learning*, where user profiles modeling the users' interests are learnt from the structured item representations. CB recommenders can be classified into three main approaches depending on how the user profile is learnt: using a non-linear approach, the k Nearest Neighbor (section 2.3.2), using a probabilistic approach, the Naïve Bayes (section 2.3.3), and using linear prediction models (section 2.3.4).

Case-based recommender systems are a subtype of CB approaches that have their origins in Case-Based Reasoning [Smyth 2007]. These techniques do not attempt to build long-term user profiles, but to directly estimate the match between a user's need and the set of options available. Case-based systems can rely on utility-based or knowledge-based technologies to assess the similarity between a case and a user's query. An extension of case-based recommenders are the conversational or critiquing-based recommenders [Ricci and Nguyen 2007], which are characterized by a reactive behavior, and are especially suitable for recommending products infrequently bought in a user's life, or those demanding a significant financial commitment, such as cars or high-end smart-phones.

2.3.1. *Item Representation*

In order that CB recommendation techniques can be applied, first the descriptions of the items must be represented in a structured way. The simplest scenario is when raw items' descriptions are already structured, using a determined set of attributes and whose possible values are known a priori. For example, a movie can be described by the cast, the director and its genres. However, this type of descriptions has to be manually annotated by content providers, and in most domains this is often not affordable, or the descriptions available are little informative. An exception is the Pandora music recommender, where songs are manually annotated by musicians with a high level of detail about the characteristics of the song.

Typically, the common scenario of CB recommendation techniques is to deal with unstructured (free-text) descriptions; thus, no attributes with well-defined values are available a priori. Some examples of such type of recommendable items are Web pages, news, e-commerce products [Chen and Sycara 1998; Billsus and Pazzani 1999; Ahn et al. 2007]. In these recommendation domains, keyword extraction techniques are necessary to obtain a structured representation of the items' content. A popular technique in IR and CB recommenders is to represent items using the Vector Space Model (VSM) and the TF-IDF (Term Frequency-Inverse Document Frequency) weighting mechanism [Manning et al. 2009]. In the VSM, given a set of k informative keywords (or attributes) $A = [a_1, a_2, \dots, a_k]$ describing the items of a recommendation domain, items are represented in a k-dimensional vector space where each element w_{ik} represents the relevance of attribute a_k in item i according to the TF-IDF weighting scheme:

$$w_i = [w_{i1}, w_{i2}, \dots, w_{ik}] \tag{2.7}$$

The TF-IDF is based on the frequency of occurrence of attributes in the item descriptions, and assumes that attributes occurring frequently in one item (Term Frequency), but rarely in the rest of items (Inverse Document Frequency), are more likely to be relevant to the description of the item. Let *I* be the set of items that can be recommended, and I_k the subset of items in which the attribute a appears, then the standard TF-IDF scheme for a given item *i* is computed as follows:

$$w_{ia} = \frac{f_{ai}}{max_{a^*} f_{a^*i}} \times \log \frac{|I|}{|I_a|} \tag{2.8}$$

where f_{ai} is the raw frequency of attribute a in the item *i*, and $max_j f_{a^*i}$ stands for the maximum of the frequencies f_{a^*i} of all attributes a^* describing item *i*.

More recently, some approaches have been proposed using user-generated annotations as item attributes [Sen et al. 2009; Nguyen and Riedl 2013]. These systems, known as tag-based recommenders or “tagommenders”, use the most popular form of user generated content, *folksonomies*, where users collaboratively annotate and categorize items with freely chosen keywords (i.e. user tags). However, due to the inherent noise of user tags, in these CB recommendation techniques it is necessary first to carefully select those tags that are useful to build informative user profiles. Additionally, given the peculiarity of this type of content, apart from TF-IDF weights other weighting schemes especially designed for user tags can be used instead [Sen et al. 2009].

2.3.2. *Nearest Neighbor: a Non-Linear Approach*

The nearest neighbor CB recommendation algorithm is usually known as a lazy learner because it simply stores in memory all the training user data in memory, and classifies a new candidate item by comparing it to the previously rated items using a similarity measure. In particular, the algorithm generates a prediction score based on the ratings of the most similar items previously rated by the user (the nearest neighbors). Depending on how the ratings are combined several estimation functions can be used. For instance, using the weighted average one can generate predictions as follows:

$$\hat{r}_{ui} = \frac{\sum_{j \in S_{iu}} s_{ij} \cdot r_{uj}}{\sum_{j \in S_{iu}} s_{ij}} \tag{2.9}$$

where S_{iu} is the set of *k* most similar items to item *i* rated by the user *u*; and s_{ij} is the similarity between item *i* and *j* based on their attribute-based profiles, which typically is computed as the cosine of the angle between their respective vector representations. Although it is the least popular CB approach in the literature, some systems like *Daily Learner* [Billsus and Pazzani 2000] have used this method to create a model of the user’s short-term interests.

2.3.3. *Naïve Bayes: a Probabilistic Approach*

The Naïve Bayesian classifier is a probabilistic classification approach that has been adopted in several CB recommender systems, such as *NewsDude* [Billsus and Pazzani 1999], *LIBRA* [Melville et al. 2002], and *ITR* [Degemmis et al. 2008]. This approach generates a probabilistic model based on the set of training ratings, which usually is employed to classify the candidate items to be recommended to a user *u* into two possible classes: I_u^+, which is the set of items known to be relevant to user *u*; and I_u^-, the set of items not relevant to user *u*. To classify a candidate item *i*, for each class the Bayes theorem is applied to calculate the corresponding *a posteriori* probabilities $P(I_u^+|i)$ and $P(I_u^-|i)$, and the class with the highest probability is selected.

$$P(I_u^+|i) = \frac{P(I_u^+)P(i|I_u^+)}{P(i)} \tag{2.10}$$

$$P(I_u^-|i) = \frac{P(I_u^-)P(i|I_u^-)}{P(i)}$$

The Naïve Bayes can also be used to generate a prediction score $\hat{r}_{ui}$ by computing the ratio of the *a posteriori* probabilities previously mentioned as follows:

$$\hat{r}_{ui} = \frac{P(I_u^+)P(i|I_u^+)}{P(I_u^-)P(i|I_u^-)} \tag{2.11}$$

where $P(i|I_u^+)$ is the probability of selecting the item *i* from the set I_u^+ (the relevant items of user *u*); $P(I_u^+)$ is the probability of selecting an item from the item collection *I* relevant to the user *u*; $P(i|I_u^-)$ and $P(I_u^-)$ are defined similarly but based on the non-relevant items of user *u*.

Empirical results show that the best performing working model of the naïve Bayes to text classification and CB recommendation is the multinomial event model, especially in data sets where the set of possible item's attributes *A* is large [McCallum and Nigam 1998]. This model uses the attribute-based item representation to estimate $P(i|I_u^+)$ and $P(i|I_u^-)$ as the product of the probabilities of the attributes that appear in the description of item *i*, assuming the attribute independence assumption:

$$P(i|I_u^+) = \prod_{a \in i} P(a|I_u^+)^{f_{ai}} \tag{2.12}$$

$$P(i|I_u^-) = \prod_{a \in i} P(a|I_u^-)^{f_{ai}}$$

Finally, a key step for implementing naïve Bayes is the estimation of attribute probabilities $P(a|I_u^+)$ that can be derived from training data as follows:

$$P(a|I_u^+) = \frac{\sum_{i=1}^{|I_u^+|} f_{ai}}{|I_u^+|} \tag{2.13}$$

$$P(a|I_u^-) = \frac{\sum_{i=1}^{|I_u^-|} f_{ai}}{|I_u^-|}$$

To make the attribute probability estimates more robust, a smoothing method is usually employed to modify the probability of infrequently attributes [Witten and Bell 1991]. More recently in Kim et al. [2006] the authors proposed a multivariate Poisson model that mitigates some limitations of the multinomial model when handling few training items available, and allows more reasonable parameter estimation under those conditions. From the learnt attribute probabilities it is also possible to build a user profile structured in two parts [Degemmis et al. 2008]: one containing the attributes that are relevant to identify items the user likes (i.e. positive interests), and other one containing the negative interests.

2.3.4. *Linear Models*

This family of CB approaches consists of inducing linear prototype vectors representing the user interests in terms of items' attributes, which then are used to recommend items whose attributes match the user's interests. To estimate the degree of matching between the user and the item profile several similarity measures can be employed, but the most common ones are the cosine similarity, and the dot product[3] between the corresponding vectors. To apply these techniques, user profiles must be represented using the same attribute-based space ($A = [a_1, a_2, \dots, a_k]$) used for representing the items. Here we denote the interest profile of a user *u* as follows (where w_{ua} indicates the degree of interest in attribute a of user *u*):

$$w_u = [w_{u1}, w_{u2}, \dots, w_{uk}] \tag{2.14}$$

One of the earliest methods using this idea was the Rocchio's method [1971], which is an adaptation of the relevance feedback technique adopted in IR to help users to incrementally refine queries based on previous search results. The adaptation for CB recommendation consists of refining the user profile instead of a query. Rocchio's method learns the degree of interest of a user *u* in an attribute a as follows:

$$w_{ua} = \alpha \cdot \sum_{i \in I_u^+} \frac{w_{ia}}{|I_u^+|} - \beta \cdot \sum_{i \in I_u^-} \frac{w_{ia}}{|I_u^-|} \tag{2.15}$$

where α and β are model meta-parameters to control the relative importance of the positive and negative examples. This approach has been used in several CB recommenders, such as *Fab* [Balabanovic and Shoham 1997], *YourNews* [Ahn et al. 2007], and [Degemmis et al. 2007].

[3] Recall that the dot product between two vectors x,y ∈ $\mathbb{R}^f$ is defined as: $x^T y = \sum_{k=1}^{f} x_k \cdot y_k$

A limitation of Rocchio's method is that it does not consider the actual values of user ratings. Therefore, recent works have proposed methods for incorporating this information in the user-profile learning process; in this thesis, we call them *rating average* methods. For example, Sen et al. [2009] propose a method of predicting the degree of interest of a user u in an attribute a as the weighted user's average rating for items that the user has rated and contain the specific attribute (here we denote this set of items as I_{ua}):

$$w_{ua} = \frac{\sum_{i \in I_{uk}} w_{ia} \times r_{ui}}{\sum_{i \in I_{ua}} w_{ia}} \tag{2.16}$$

Finally, a more sophisticated user-profile weighting method consists of modeling the user's interests as part of a linear regression model, in which ratings are expressed as a linear combination of item and user profiles (i.e. attribute weights). Recently, it has been demonstrated that this method, when applied to tag-based recommendation, can perform similarly to state-of-the-art CF techniques in terms of prediction accuracy [Nguyen and Riedl 2013].

An efficient way to implement this method is by using the SGD approach previously introduced in Section 2.2. In this case, incorporating also the baseline predictor b_{ui}, the CB rating prediction model predicts a rating for a given user and item as the dot product of the corresponding attribute-based user and item profiles:

$$\hat{r}_{ui} = \mu + b_u + b_i + w_i^T w_u \tag{2.17}$$

Then, the interest profiles w_u can be incrementally learnt together with the user and item biases during the minimization of the following objective function:

$$\min_{b_*, w_*} \sum_{r \in R} \left[\left(r_{ui} - \mu - b_u - b_i - w_i^T w_u \right)^2 + \lambda \left(b_u^2 + b_i^2 + \|w_u\|^2 \right) \right] \tag{2.18}$$

Similarly to the case of the baseline parameters, for a given training rating r_{ui}, each parameter w_{ua} is modified in the opposite direction of the function's gradient, which corresponds to the prediction error e_{ui} times the item's vector (w_i), as follows:

$$w_u \leftarrow w_u + \gamma (e_{ui} \cdot w_i - \lambda \cdot w_u) \tag{2.19}$$

2.3.5. *Limitations of Content-Based Recommendation Techniques*

CB recommenders suffer from three main limitations that affect their effectiveness [Lops et al. 2011]:

- **Limited content analysis**. The effectiveness of CB techniques strongly depends on the quality of the content analysis process, in which structured representations of items are derived from raw item descriptions. If the generated item representations do not contain distinguishing aspects of the items to discriminate the ones the user likes or does not like, these approaches cannot provide accurate predictions.

- **Over-specialization**. Because the estimations of CB approaches are based on the good match between the attributes of the target item and the interests of the target user, they tend to produce recommendations with a limited degree of diversity. In other words, all the items suggested will share some attributes, especially if the user profile contain few interests.
- **Requirement of a minimum number of user's ratings (new-user cold-start)**. CB approaches require a minimum number of user's ratings in order to build an accurate profile of the user's interests. Users that have rated fewer items than this minimum are usually referred to as new users when using the wider sense of the cold-start problem [Schein et al. 2002].

2.4. Collaborative Filtering

Collaborative Filtering (CF) techniques [Ekstrand et al. 2010; Koren and Bell 2011] base their predictions on the ratings of other users. Differently from CB recommenders, CF approaches are domain-independent in the sense that no item descriptions are required in order to generate recommendations, only ratings. This technique emulates a simple but effective social strategy called "word-of-mouth", which relies on the added credibility of person-to-person recommendations. The basic assumption behind CF is that a user may be interested in items that have been positively rated by other users with similar interests.

In the literature, three main CF techniques can be identified: *user-based* CF (Section 2.4.1), which was the first one proposed and directly implements the basic assumption of CF by computing the *k* nearest neighbors of a target user based on their rating's behavior similarity; *item-based* CF (Section 2.4.2), which is a more scalable implementation than the previous one where, rather than using similarities between users' rating behavior, it employs similarities between the items' rating pattern; and *Matrix Factorization (MF)* (Section 2.4.3), nowadays the most popular CF approach because of its superior prediction accuracy and scalability compared to *nearest neighbors* methods, which represents items and users as vectors of latent factors directly inferred from the rating matrix, and generate predictions by directly comparing the latent factors of users and items.

2.4.1. *User-based CF*

User-based CF is a straightforward implementation of the CF's core idea: find top *k* users whose past rating behavior is most similar to that of the target user (i.e. nearest neighbors) and compute a rating prediction for the target user and item based on the ratings provided to that item by the user's neighborhood. Well-known examples of recommender systems using the user-based CF approach are the *GroupLens Usenet* [Resnick et al. 1994; Konstan et al. 1997], *Ringo* [Shardanand and Maes 1995], and *BellCore* [Hill et al. 1995].

The most common method to generate a prediction for a certain user *u* and item *i* using user-based CF consists of computing a weighted average of the normalized ratings for item *i* provided by the neighbors of user *u* according to their user-to-user similarity values:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_u} s_{uv}(r_{ui} - b_{vi})}{\sum_{v \in N_u} s_{uv}} \tag{2.20}$$

where: N_u is the user's neighborhood (i.e. top *k* most similar users to user *u*) and s_{uv} is the user-to-user similarity value. The baseline predictors b_{vi} are used to adjust the ratings provided by neighbor users subtracting the corresponding user and item biases.

A key design decision when implementing user-based CF is the choice of the similarity measure to compute user-to-user similarities based on their rating correlation. In the literature, several measures have been proposed, such as the Spearman rank correlation [Herlocker et al. 2002] and the cosine similarity, but the most popular one is the Pearson correlation, which computes the statistical correlation (Pearson's r) between two user's common ratings to determine their similarity, and is usually computed as follows:

$$s_{uv} = \frac{\sum_{i \in I_u \cap I_v}(r_{ui} - b_{ui})(r_{vi} - b_{vi})}{\sqrt{\sum_{i \in I_u \cap I_v}(r_{ui} - b_{ui})^2}\sqrt{\sum_{i \in I_u \cap I_v}(r_{vi} - b_{vi})^2}} \tag{2.21}$$

Again, the baseline predictors are used to remove from the ratings the particular user and item rating biases. In order to avoid high similarity values between users with few ratings in common, a damping method is commonly applied to reduce the similarity in those cases. Basically, two methods for damping can be used: one consists of reducing the similarity only when the number of co-rated items is lower than a specific threshold; and the other one consists of incorporating a damping term to the denominator, which always reduces the similarity but especially when users have a small number of co-rated items.

Another important design decision is the choice of the size of the user's neighborhood, a data-dependent meta-parameter that should be obtained experimentally. This parameter controls the trade-off between the generalization power of the prediction model (the more users, the more reliable may be the prediction) and the noise introduced by those users with low similarity to the target one.

2.4.2. *Item-based CF*

Item-based CF is another nearest neighbor variant that, similarly to the CB approach presented in Section 2.3.2 (Eq. 2.9), computes a rating prediction based on the weighted average of the ratings given by the user to the items most similar to the target item. But the key difference between the two methods is that in the CF approach the similarity is based on user's rating behavior rather than item attributes. Hence, incorporating also the user and item biases the rating estimation function commonly used in item-based CF is the following:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in S_{iu}} s_{ij}(r_{uj} - b_{uj})}{\sum_{j \in S_{iu}} s_{ij}} \qquad (2.22)$$

As in user-based CF the choice of how to measure the rating-based item-to-item similarity is a critical design decision of item-based CF. In this case the most popular similarity metric is the *cosine* but other measures have been also proposed, such as an adaptation of the Pearson's correlation used in user-based CF [Sarwar et al. 2000], and conditional probabilities [Karypis 2001].

Due to the relatively static nature of item similarities in domains where the number of users is much larger than the number of items (i.e. $|U| \gg |I|$), especially when there are more ratings per item than per user, it is possible to pre-compute the item-to-item similarities and thus perform predictions much faster than in user-based CF. For this reason, item-based CF has been adopted in several e-commerce websites with a large user base, like for instance in *Amazon*'s recommender [Linden et al. 2003].

2.4.3. *Matrix Factorization*

MF prediction models use Singular Value Decomposition (SVD) to reduce the dimensionality of the original rating matrix to a latent factor space, which characterizes those aspects of items and users that are relevant for both describing the item and modeling the user's interest. Once the latent factor model is learnt, recommendations are produced based on the dot product of the target user and item factor vectors.

Figure 2.1 illustrates how SVD decomposes the original *m*×*n item-to-user* rating matrix into three smaller matrices ($A = U \times S \times V^T$) with *U* an *m*×*r* matrix, *S* a *r*×*r* diagonal matrix, and *V* an *n*×*r* matrix, such that *U* and *V* are orthogonal and *r* is the rank of the original matrix *A*. Then, the best rank-*k* approximation of *A*, which is denoted with A_k, is produced by selecting the *k* largest singular values in *S* and set the others to zero (i.e. $A_k = U_k \times S_k \times V_k^T$).

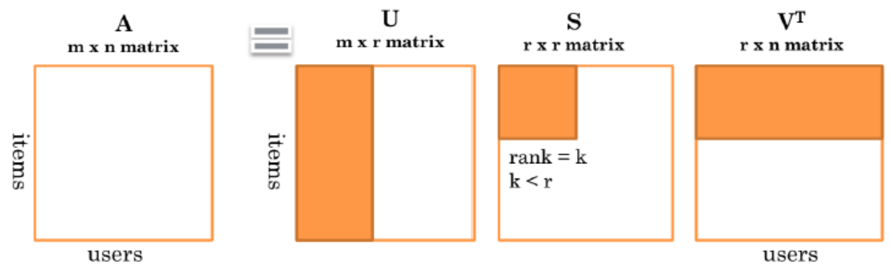


Figure 2.1. Low-rank matrix approximation using SVD (Figure from B. Mobasher UMAP 2013)

Due to the typical high number of missing values (i.e. high sparsity) of the rating matrix, it is not practical to apply the conventional SVD method, widely used in IR to solve the problems of synonymy and polysemy, since it requires a full dense rating matrix [Deerwester et al. 1990]. An early solution to this limitation consisted of filling missing values with pseudo-generated ratings [Sarwar et al. 2000], but this strategy is usually not accurate and, in addition, it does not scale well. A more accurate and efficient method is to use an implementation of the *Lanczos* algorithm based on Expectation Maximization (EM) framework, which is designed to work with sparse matrices. However, the learning process is still costly when applied to large matrices, like in the Netflix prize data set [Kurucz et al. 2007]. For this reason, most researchers using MF prediction models have focused on learning the latent factors based only on the known training ratings while avoiding over-fitting using regularization of the model parameters. A popular method to learn the user and item latent factors in this way is by using the SGD optimization method popularized by Funk [2006] (described in Section 2.2), which was widely used during the Netflix prize because of its efficiency and efficacy, and posteriori used as part of the winning solutions [Koren and Bell 2011]. **Figure 2.2** illustrates the reduction dimensionality process when using the SGD Funk's method. In this case, the rating matrix A is directly decomposed as the dot product between the corresponding user and item vectors of latent factors $p_u, q_i \in \mathbb{R}^f$ over the known training ratings.

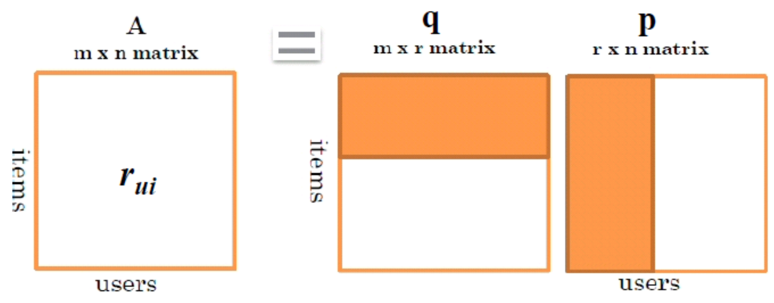


Figure 2.2. Low-rank approximation using non-conventional SVD (Figure from B. Mobasher UMAP 2013)

Incorporating the baseline predictors based on user and item rating biases, the standard MF prediction model (also known as *bias MF*) computes the rating prediction for a given user *u* and item *i* as the sum of the user and item biases and the dot product between their corresponding vectors of latent factors:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{2.23}$$

Considering this MF rating prediction model, the parameters can be optimized for rating prediction task by minimizing the least square objective function as follows:

$$\min_{b_*q_*p_*} \sum_{r \in R} \left[\left(r_{ui} - \mu - b_u - b_i - q_i^T p_u \right)^2 + \lambda \left(b_u^2 + b_i^2 + \|q_i\|^2 + \|p_u\|^2 \right) \right] \tag{2.24}$$

Using the SGD method, for a given training rating r_{ui} the user and item factors are modified in the opposite direction of the gradient (i.e. the error e_{ui}times the item or user factor vectors):

$$q_i \leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda \cdot q_i) \tag{2.25}$$

$$p_u \leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda \cdot p_u)$$

During the Netflix competition, it was demonstrated that MF models are the best CF approaches in terms of prediction accuracy and scalability, especially on large and sparse data sets like the movie data set of ratings that was provided for the contest. For this reason, MF prediction models have gained popularity in the last years on the research community. Koren [2008] proposed a variant of *bias MF* (known as *SVD++*) that improves the prediction accuracy by considering also the implicit feedback (i.e. unary ratings) provided by the users. In particular, *SVD++* adds an additional set of latent factors relating each item *i* to a factor vector $y_i \in \mathbb{R}^f$, which are used to characterize users based on the set of items that they rated (i.e. here the action of rating an item is considered as implicit feedback). The extended MF prediction model is defined as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\alpha} \sum_{j \in I_u} y_j \right) \tag{2.26}$$

where α is a normalization factor that stabilizes the variance of the sum of factors across the range of observed values of I_u, since they are centered around zero by the regularization. The model parameters are also learnt by minimizing the regularized squared error objective function using SGD.

Rather than targeting the rating prediction task, recent research started to exploit possibilities for ranking-oriented MF approaches that focus on the quality of recommendation or ranked item lists, which is especially interesting in scenarios where only implicit feedback is available [Weimer et al. 2009]. For example, Rendle et al. [2009] proposed a MF approach dealing with implicit feedback based on the Bayesian Personalized Ranking (BPR-MF) framework, which is directly optimized for a smoothed ranking objective function based on the Area Under the ROC Curve (AUC), a well-known IR performance metric. Essentially, this technique considers entity pairs instead of single entities in its objective function, allowing the interpretation of unary ratings as partial ranking data. The objective function of BPR-MF is defined as follows:

$$\min_{q_*p_*} \sum_{(u,i,j) \in D_S} \left[\ln \sigma\left(q_i^T p_u - q_j^T p_u\right) + \lambda(\|q_i\|^2 + \|p_u\|^2) \right] \tag{2.27}$$

where $D_S = \{(u,i,j) | i \in I_u^+ \wedge j \in I_u^-\}$ constitutes the partial ranking data, which are training triples randomly chosen and uniformly distributed, used for training the model parameters through SGD; and σ is the logistic function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.28}$$

Collaborative Less-is-More Filtering (CLiMF) [Shi et al. 2012b] is a ranking-oriented approach similar to BPR-MF, but that optimizes a smoothed version of the Mean Reciprocal Rank (MRR) (another ranking accuracy metric) rather than AUC. The main advantage is that it uses a top-biased measure: errors at lower ranked positions of the recommendation list are more penalized than higher ranked positions, which is the desired goal when recommending ranked item lists.

2.4.4. *Limitations of Collaborative-Filtering Recommendation Techniques*

CF recommenders suffer from several limitations that affect their effectiveness [Adomavicius and Tuzhilin, 2005]:

- **Data Sparsity**. In most recommendation domains the rating matrix is highly sparse, in other words, the number of ratings already obtained is usually very small compared to the number of missing ratings (i.e. the ones the system has to predict). Because CF techniques rely on general rating patterns across users and items, their accuracy strongly depends on the rating matrix sparsity, especially in *nearest neighbors* CF approaches. MF prediction models usually perform better on sparse data sets, since they are based on rating patterns derived from a reduced and more informative representation of the rating matrix.
- **Requirement of a minimum number of user's ratings (new user cold-start).** Similarly to CB techniques, CF prediction models require a minimum number of ratings per user in order to produce acceptable personalized recommendations.
- **Requirement of a minimum number of item's ratings (new item cold-start).** Additionally to the new user problem, CF approaches cannot produce accurate recommendation for items that have been rated by few users, since no content-based information is exploited in these approaches.
- **Lack of scalability**. The computational cost of neighborhood CF approaches grows exponentially with the number of users (user-based CF) and the number of items (item-based CF). For this reason, the application of these techniques in systems that deal with millions of users or items can be impractical. In those cases, MF prediction models are a better option because they build predictive models that can work in a much lower-dimensional space compared to the number of users and items.

2.5. Context-Aware Recommender Systems

Context-Aware recommender systems (CARSs) extend the traditional formulation of the recommendation problem by incorporating also the context of user-item interactions. Therefore, CARSs estimate the rating of a target user u for an item i, not only based on a data set of ratings (of users for items), but they also exploit both the contextual information under which the ratings were acquired and the contextual situation of the target user asking for a recommendation. More formally, given a set of possible contextual situations S that users can encounter while experiencing items, the rating estimation function is formulated as $\hat{r}: U \times I \times S \rightarrow R$ and estimates the rating of a given user for a given item in a specific contextual situation.

The main assumption supporting CARS techniques is that in many domains users can experience items differently depending on their contextual situation, and thus user's ratings can also be influenced by the context. For instance, depending on the weather, a user may prefer to visit a beach or a museum. Therefore, CARSs' goal is to enhance the effectiveness of recommendations by incorporating context into the recommendation process. Depending on how context is exploited, three main types of contextual paradigms can be identified [Adomavicius and Tuzhilin 2011; Adomavicius et al. 2011]: contextual *pre-filtering* (Section 2.5.2), where context is used for selecting the relevant set of rating data before computing predictions with a context-free model; contextual *post-filtering* (Section 2.5.3), where context is used to adjust predictions generated by a context-free model; and *contextual modeling* (Section 2.5.4), in which contextual information is directly incorporated in a context-free model as additional parameters. Before describing these contextual paradigms, in Section 2.5.1 we show the main views of understanding and representing contextual knowledge in CARSs.

Earlier research on CARS focused on following a context-driven querying and search approach, such as *INTRIGUE* [Ardissono et al. 2003] and *COMPASS* [Van Setten et al. 2004]. These systems usually do not attempt to model users' contextual preferences, but only to exploit the current context as queries to search for the most appropriate items.

2.5.1. *Context Representation*

Context is a multifaceted concept used across various disciplines, each one taking its own perspective of the concept. The definition of Dey et al. [2001] was probably the first broadly adopted in AI and ubiquitous computing: "*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*". In CARSs, context characterizes the situation of a user, an item, and the experience the user is evaluating. Dourish [2004] introduces a taxonomy of contexts according to which contextual information can be classified into the interactional and the representational views.

The *interactional* view assumes that the user behavior is induced by an underlying context, but the context itself is not necessarily observable. Therefore, no enumeration of contextual conditions is possible beforehand, since the scope of contextual information is defined dynamically. This view assumes the existence of a cyclical relationship between context and the activity: context gives rise to the activity and the activity changes the context. Anand and Mobasher [2007] proposed a generic interaction framework for CARS inspired by the Atkinson and Shriffin's model of human memory. This framework emphasizes the distinction between local preference models in short-term memory and global, static long-term memory, and the relevance of user's interaction with the system in deriving contextual cues, which can be classified into three types: collaborative, semantic and behavioral. Some works have proposed implementations based on this framework: in Sieg et al. [2007] context is reflected in the current interests of the user during her current interaction with the system, and in Hariri et al. [2012] the authors proposed a context-aware music recommender based on latent topic sequential patterns, where context is reflected in the sequence of songs played by a user during her active interaction with the system. Other works have also exploited runtime context to enhance predictions [Hayes and Cunninghamin 2004; Cantador et al. 2008].

The *representational view* assumes that context is defined with a predefined set of observable conditions, which can be separated from the activity and the structure of which does not change significantly over time. In other words, using this view a set of contextual factors and their conditions are identifiable and known a priori and, therefore, can be represented beforehand. Similarly as in the structured item representations in CB (see Section 2.3.1) a contextual situation $s \in S$ is typically represented as a set of static contextual factors (e.g. *weather* and *mood*) whose possible values (i.e. contextual conditions) are known a priori (e.g. *sunny*, *cloudy* and *rainy* are typical conditions of the *weather* factor):

$$s = [c_1, c_2, \dots, c_k] \tag{2.29}$$

where, *k* is here the number of contextual factors that are captured in the application domain; and c_k is a specific contextual condition of the k-th factor, which can be also unknown if the information is not available.

An important issue in the *representational view* is the selection of the relevant contextual factors for the specific domain, since contextual information that does not have a significant contribution to explain the variance in the ratings could degrade the prediction by adding noise. Two main approaches can be used to determine whether a specific piece of contextual information should be captured and exploited during the recommendation process: (1) user surveys [Baltrunas et al. 2012], where users are explicitly asked if a specific contextual condition would positively or negatively influence their decision/evaluation (2) using statistical testing methods based on existing training data [Odić et al. 2013]. However, each of these techniques has its drawbacks: survey assessment requires a lot of user effort, and statistical testing is not reliable unless the data set is dense (i.e. items have been rated several times in different contextual situations).

2.5.2. *Pre-Filtering*

Pre-filtering is the most popular contextual paradigm because it has a straightforward justification: when context matters, use in the recommendation process only the data acquired in the same contextual situation of the target user, because only this data are relevant for predicting user preferences. As show in **Figure 2.3**, in contextual pre-filtering the idea is to use context for data pre-processing, by discarding rating data, which are not relevant for the target situation, and use the remaining ratings to learn a local model for rating prediction and recommendation.

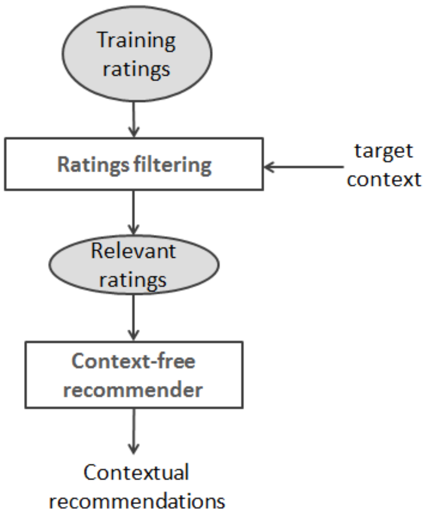


Figure 2.3. Contextual pre-filtering paradigm

Adomavicius et al. [2005] proposed a straightforward method to implement contextual pre-filtering known as the reduction-based approach. A variant of this approach is *Exact Pre-filtering*, which strictly implements the idea of building a local context model, which is tailored for each target situation, by using only the ratings tagged with that situation. Adomavicius et al. [2005] proposed to use *Exact Pre-filtering* in combination with user-based CF, also used in [Panniello et al. 2009; Campos et al. 2013], and Lombardi et al. [2009] used it in combination with a Naïve Bayes CB prediction model. The main limitation of *Exact Pre-filtering* is its rigidity when building the local models: it never reuses ratings acquired in situations syntactically different to the target one, independently from the number of training ratings available for learning the local model (i.e. it always builds a strict local model for each target contextual situation). Therefore, this approach only works for those contextual situations with enough ratings to build a robust local prediction model.

A different approach was introduced in Baltrunas and Ricci [2009; 2014], which is known as *Item Splitting*. The idea here is to split the rating vector of a given item into two virtual item vectors using a specific contextual factor. For instance, the ratings of a music track may produce two sets of ratings: the ratings of the music track collected when the user was *happy*, and the ratings of the same track when the user was *sad* (assuming that *happiness* is the contextual factor). Then, a predictive model is trained by considering all the ratings organized in the extended set of items generated by splitting all the items that satisfy a given statistical test (i.e. measuring if the two virtual items generated by the splitting are significantly different). In *Item Splitting*, filtering is selectively carried out item by item for the most relevant contextual condition. Baltrunas and Amatriain [2009] proposed a variant of this approach, which is called *User Splitting*, where instead of splitting items, it splits users into several sub-profiles, each representing the user in a particular context. Then, similarly to the previous approach, a global predictive model is built using all the ratings but in the modified set of users. Recently, Zheng et al. [2013a] also explored a combination of the two previous variants, *UI Splitting*, which yielded better prediction accuracy in a movie-rating data set. All these variants employ statistical methods (e.g. chi-square test, t-test, etc.) as splitting criteria.

Finally, Zheng et al. [2013b] presented *Differential Context Relaxation* (DCR), another type of pre-filtering modeling approach that tries to break down a predictive model into different functional components to which specific optimal contextual constraints are applied in order to maximize the performance of the whole algorithm. The authors used this approach in combination with the nearest neighbor CF models (i.e., user-based and item-based CF) because their prediction functions can be easily decomposed into several components. The selection of the relevant factors per component is performed by using a particle swarm optimization algorithm.

2.5.3. *Post-Filtering*

In contrast to pre-filtering, the post-filtering paradigm ignores the contextual information during the prediction generation. As shown in **Figure 2.4.,** here context is used only to adjust the resulting predictions made by a context-free prediction model. The basic idea of this contextual paradigm is to analyze the preference data for a given user in a given context to find specific item usage patterns (e.g., *Jane* watches only comedies on *workdays* and action movies on *weekends*) and then use these patterns to adjust the rating predictions or the ranked item list. As in pre-filtering, post-filtering has the major advantage that it allows using any of the available context-free recommendation techniques proposed in the literature.

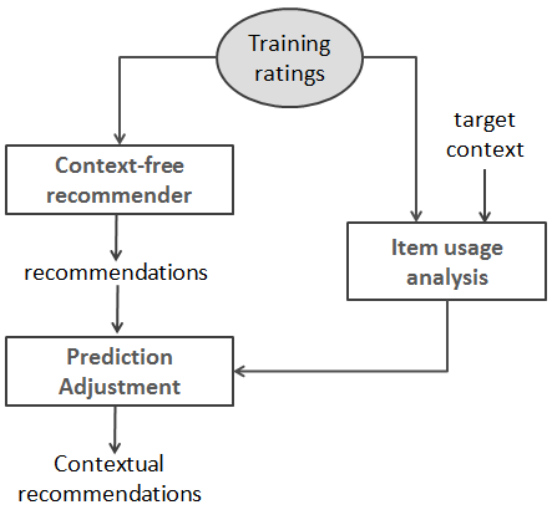


Figure 2.4. Contextual post-filtering paradigm

Two main post-filtering approaches can be identified:

- Methods that focus on finding common item features for a given user in a given context (e.g., preferred actors to watch in a given context), and then use these attributes to adjust the recommendations. An example of such an approach is *Smart Radio* [Hayes and Cunningham 2004]. In this case the system uses the genres and artists of the recent played songs as context (i.e. session-based context) to refine the recommendations based on CF, by promoting those items of the list with similar attributes.
- Methods that use a predictive model that approximates the probability of an item *i* to be relevant for the user *u* in a given context, to penalize those items not relevant to the target context. Panniello et al. [2009] proposed a simple method to estimate this probability of relevance, using user-based CF, as the proportion of users in the neighborhood of user *u* that rated the item *i* in the target situation. The authors presented two variants: *Weight Post-filtering*, which reorders the recommended items by weighting the predicted ratings with their estimated probability; *Filter Post-filtering*, which discards the recommended items that have a probability smaller than a specific threshold.

2.5.4. *Contextual Modeling*

Approaches based on contextual modeling commonly extend a context-free prediction model by explicitly modeling the influence of context on the rating prediction function, i.e., with additional parameters that represent the contextual information. This gives rise to truly *Multidimensional* (MD) recommendation algorithms, as shown in **Figure 2.5.**.

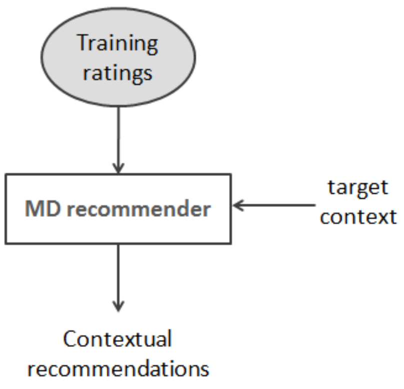


Figure 2.5. Contextual modeling paradigm

Most research on MD recommenders have focused on extending MF prediction models, although some works also have proposed variants extending user-based CF [Chen 2005; Adomavicius and Tuzhilin 2011] and CB recommendation techniques [Campos et al. 2013]. Currently, two main approaches based on extending MF prediction model have been proposed in the literature: *Tensor Factorization* (TF) and *Context-Aware Matrix Factorization* (CAMF).

TF consists of extending the two-dimensional MF problem into a multi-dimensional version, where the rating tensor is factored into a lower-dimensional vector space. In this way, the interactions between users, items, and contextual factors are represented as latent factor vectors. Several authors have proposed variants of this approach: some optimized for rating prediction, such as *Multiverse Recommendation* [Karatzoglou et al. 2010] and *Factorization Machines* [Rendle et al. 2011], and others optimized for ranking using implicit user feedback, such as *iTALS* [Hidasi and Tikk 2012], and *TFMAP* [Shi et al. 2012a]. The main limitation of these approaches is the computational complexity, since the number of model parameters grow exponentially with the number of contextual factors.

CAMF is a more scalable approach, proposed by Baltrunas et al. [2011b; 2012], since it uses less parameters than TF. The approach is a generalization of the time-aware MF model proposed by Koren [2010], an extension of *SVD++* (described in Section 2.4.3) that incorporates the temporal dynamics associated to rating data. The authors demonstrated that this prediction model, called *timeSVD++*, was one of the best performing isolated rating prediction models in the Netflix data, and indeed, a variant of the model form part of the winning solution based on an ensemble of several prediction models [Koren 2009]. In particular, this model adds time-sensitive user and item biases $b_u(t) + b_i(t)$ that change over time, and models the user factors also as a function of time $p_u(t)$, leading to the following prediction model:

$$\hat{r}_{uit} = \mu + b_u(t) + b_i(t) + q_i^T\left(p_u(t) + |I_u|^{-\alpha}\sum_{j\in I_u} y_j\right) \tag{2.30}$$

Here the user bias captures two temporal effects using the function:

$$b_u(t) = b_u + \alpha_u \cdot |t - t_u|^{\beta} + b_{u,t} \tag{2.31}$$

where b_u is the static part of the user bias; $\alpha_u \cdot |t - t_u|^{\beta}$ is a linear model for approximating a gradual drifting behavior, being t_u the user's mean date of rating, and $|t - t_u|$ the number of days between dates t and t_u; $b_{u,t}$ captures session-specific variability, modeling short-lived effects, such as a different user's *mood* that day.

The item bias captures a less fined-grained temporal variability using the following function:

$$b_i(t) = b_i + b_{i,Bin(t)} \tag{2.32}$$

where b_i is the stable part of the item bias; $b_{i,Bin(t)}$ captures temporal variability modeled by time-based bins of the same size. As the other parameters of the time-aware model, the optimal size is a data-dependent parameter that must be optimized experimentally. (E.g., in the Netflix data the authors used bins of two months.)

Baltrunas et al. [2011b; 2012] proposed to extend the static baseline predictors of MF with contextual information, modeling the interaction between items and contextual conditions, but without considering temporal dynamics. The authors demonstrated that this solution outperforms the TF implementation proposed by Karatzoglou et al. [2010] in small-medium size data sets. Baltrunas et al. [2011b; 2012] proposed three different variants of CAMF that model the influence of contextual conditions at different granularities: CAMF-C models the influence of a condition globally, i.e., assuming that it has the same effect on every user and item; CAMF-CI models the influence of a contextual condition uniformly on each item, i.e., assuming that it does not depend on the user; and CAMF-CC, which assumes that context influences uniformly the ratings for all the items of the same type (i.e. based on item categories). For example, using CAMF-CI the contextual rating prediction function is defined as follows:

$$\hat{r}_{uic_1 \ldots c_k} = \mu + b_u + b_i + \sum_{j=1}^{k} b_{ic_j} + q_i^T p_u \tag{2.33}$$

More Recently, Odić et al. [2013] also experimented with a variant of CAMF that models the influence of contextual conditions with respect to the users. This variant, called CAMF-CU has the following estimation formula:

$$\hat{r}_{uic_1 \ldots c_k} = \mu + b_u + b_i + \sum_{j=1}^{k} b_{uc_j} + q_i^T p_u \tag{2.34}$$

2.5.5. *Limitations of Context-Aware Recommender Systems*

CARS are based on CB and CF approaches and thus they suffer from the same limitations. Actually, the **sparsity** and **cold-start** problems are aggravated, especially in pre-filtering approaches, because they require for a large data set of contextually tagged ratings, i.e., ratings for items provided in various contextual situations that may be encountered by a user while experiencing an item, in order to accurately identify the user's interests in each of the possible contexts. The **scalability** of CARS can also be a problem, especially in TF approaches, if the number of contextual factors is relatively large.

In addition to the previous limitations, the effectiveness of CARS following the *representational view* strongly depends on the **relevance of contextual factors and its conditions** that are acquired and represented in the system. Similarly to CB recommenders, if the generated representations do not contain relevant contextual conditions describing ratings' contexts that help to identify different user's behaviors on the same kind of items, these approaches are not able to improve context-free recommendations.

2.6. Hybrid Recommender Systems

As in other machine learning applications, the highest prediction accuracy for a particular recommendation domain is usually achieved by combining different prediction models. This was clearly demonstrated during the Netflix prize, where the most accurate solutions employed a combination of more than one hundred prediction models using a specific ensemble scheme. (The winning solution of the Netflix prize used Gradient Descent Boosted Trees [Koren 2009].) However, in typical recommender systems it is not feasible to combine such a high number of models, mainly because it is expensive to maintain this kind of ensembles over time. For this reason, practical solutions consist of combining fewer prediction models.

Two main advantages of CB approaches complement well with the sparsity-related limitations of CF: (1) the user independence, since only the ratings provided by the target user are exploited to build her own profile; (2) recommendation of new items, since CB approaches also exploit the content information of the items and not only the ratings associated to them. For this reason, the most common hybrid strategies consist of combining CB and CF approaches. Burke [2002; 2007] presents a classification of such hybrid strategies depending on how the combination is made. In the literature, the most popular strategies are the feature augmentation/combination (also known as *content-boosted CF*) and the meta-level hybrid (also known as *collaboration through content).*

Finally, another interesting alternative is to combine different context-aware paradigms (i.e. pre-filtering, post-filtering or contextual modeling methods), since the utility of contextual factors may be different depending on whether it is used for contextual pre-filtering, post-filtering or contextual modeling. For example, time information (*weekday* versus *weekend*) may be most useful to pre-filter relevant data, but weather information (*sunny* versus *rainy*) may be the most appropriate to use as a post-filter. *Cinemappy* [Ostuni et al. 2012], is a context-aware movie recommender that employs such a combination based on the utility of different contextual factors. In particular, the *companion* factor is exploited by a *User-Splitting* pre-filtering approach, which produces several profiles for the same user depending on the user companion, and the *location* factor is exploited by a post-filtering method that re-ranks the recommendation list based on geographic criteria.

2.6.1. *Feature Augmentation/Combination Strategy*

Feature augmentation is a strategy for hybrid recommendation in which a contributing component produces a new set of features that then are part of the input to the another recommendation technique (the primary component). This strategy is usually employed when the strong primary recommendation component is a CF approach, and the goal is to boost its performance by exploiting content-based features, which can be generated by means of a CB approach. For this reason, approaches using this strategy are also known as *content-boosted CF*.

A variant of this strategy consists of filling the missing ratings of the rating matrix by using a CB technique. For example, Melville et al. [2002] employed a Naïve Bayes CB to predict rating predictions used to fill the missing ratings of the rating matrix, and then train a user-based CF with the dense pseudo ratings matrix.

More recent solutions have focused on combining content-based features into the logic of CF techniques. For instance, Mobasher et al. [2003] and Lees-Miller et al. [2008] estimate *item-to-item* similarities based on item attributes to boost an item-based CF by means of a linear combination with the rating-based similarities. Gunawardana and Meek [2008] proposed a CF technique based on Boltzmann machines, a type of latent factor models using neural networks, whose parameters are constrained according to the content associated with the items, allowing the model to use content information to recommend items that are not rated during training (i.e. cold-start items).

Finally, some approaches have also incorporated content-based features to improve the performance of MF models. Forbes and Zhu [2011] proposed a variation of the standard MF in which q_i the factor vectors of item i are constrained to depend explicitly on their item features (w_i), formulating the rating prediction model as follows:

$$\hat{r}_{ui} = p_u^T O^T w_i \tag{2.35}$$

Where O stands for a *feature-by-factor* matrix representing the latent factor vectors of the items' attributes. An interesting characteristic of this approach is that this latent representation of item attributes can be also used to measure distributional similarities between attributes based on the user preferences, by comparing the generated latent factor vectors after training the MF model (see Section 3.2.2 for more details about distributional semantics).

Katz et al. [2012] proposed a combination of two methods to boost MF models with Wikipedia content: (1) creating pseudo-ratings using a nearest neighbor CB prediction model, and (2) incorporating item-to-item similarities by adding additional latent factors that are weighted according to the similarity values. Finally, Shi et al. [2013] also incorporate *item-to-item* content-based similarities into a standard MF model, but in this case, by means of a joint MF model, in which additional cost functions are incorporated in order that similar items share also similar latent factors.

2.6.2. *Meta-Level Strategy*

Differently from feature augmentation, in *meta-level* strategy the contributing component produces a completely new model based on training rating data, and not only additional features, which is then used for training the primary component. When combining CB and CF, this strategy usually consists of first building interest profiles by using a CB technique and then using the generated user profiles to estimate the *user-to-user* similarities in a user-based CF algorithm. For this reason, this hybrid strategy is also known as *collaboration through content*.

In the literature, several works have followed the meta-level strategy using different linear CB models for learning the user profiles: the Rocchio’s algorithm [Balabanovic and Shoham 1997; Degemmis et al. 2007], the Winnow method [Pazzani 1999], and *rating average* methods [Liu et al. 2007; Sieg et al. 2010].

Chapter 3 - Semantically-Enhanced Recommender Systems

3.1. Introduction

Semantically-Enhanced Recommender Systems (SERSs) aim at improving the effectiveness of previously mentioned recommendation techniques by exploiting semantic relationships among *concepts;* these concepts are used in CB techniques or hybrid recommenders to describe items (i.e. items' attributes) and in CARSs to describe contextual situations (i.e. contextual conditions).

The goal of SERSs is then to exploit semantics that is not exploited by CB techniques using traditional representations (see Section 2.3.1) and by CARSs using the representational view of context (see Section 2.5.1). Traditional representations only rely on "syntactic" evidence of concept relatedness and this limits their accuracy, especially when rating data are sparse. For example, in a CB recommender of art works, if the system only exploits the knowledge that the user is interested in *Claude Monet's* paintings, it will be able to recommend paintings of the same artist, but not paintings of other artists like *Pierre-Auguste Renoir*, even though they are likely to be also relevant for the user because they are semantically related (both are *Impressionist* paintings). The main assumption in SERSs is that, in a recommendation domain, one can identify domain-specific semantic similarities between concepts describing items or contextual situations, which can be useful to mitigate the sparsity-related limitations and thus enhance the performance of recommendation techniques based on concept matching.

This chapter is organized as follows. Section 3.2 presents the two major families of semantic similarity measures: the ontology-based measures and the distributional measures. Section 3.3 reviews efforts focusing on improving existing CB techniques by exploiting attribute-to-attribute semantic similarities. Section 3.4 presents the experimental evaluation we performed to a specific SERS technique for enhanced CB recommendation using the Netflix prize data set. Finally, Section 3.5 reviews some of the existing CARS techniques that exploit domain-specific situation-to-situation semantic similarities in the context-modeling phase to enhance their performance.

3.2. Semantic Similarity of Concepts

The semantic similarity of two concepts is measured differently depending on the knowledge source from which associations are derived. In the literature, two main types of semantic similarity measures can be identified: *ontology-based* measures, which estimate similarities based on the explicit, formal structure of ontologies; and data-driven *distributional* measures, where semantic associations are directly derived from a corpus of data.

The main goal of both type measures is to mimic human judgment of semantic relatedness between a given pair of concepts. Typically, these measures have been used in numerous Natural Language Processing tasks, such as word sense disambiguation, spelling correction, text summarization and also IR. In the following subsections, we present the major ontology-based and distributional similarity measures used by some SERSs.

3.2.1. *Ontology-based Measures*

Ontology-based similarity measures exploit the structure of the ontology in order to infer an estimation of how semantically related two concepts are. In the field of AI, a widely accepted definition of ontology is the one provided by Studer et al. [1998]: "*An ontology is a formal, explicit specification of a shared conceptualization*". Depending on the degree of complexity of the specification, ontologies can be classified (from lower to higher complexity) into: thesauri, like for instance *WordNet* [Miller 1995]; is-a hierarchies or taxonomies (e.g., the *Open Directory Project* (ODP) [http://www.dmoz.org/]), ontologies with domain-specific types of relations (e.g., the *DBpedia* [http://dbpedia.org/]), and ontologies with logical and value restrictions.

In practice, given the limited availability of expressive domain-specific ontologies, most of the existing concept-to-concept ontology-based measures only exploit hierarchy-based relations, and have been especially designed for being used with *WordNet* (for this reason they are also known as *WordNet*-based measures). Basically, three different types of hierarchy-based measures are identified in the literature: link-based measures, node-based and hybrids combing both measures.

Link-based (or edge-based) measures are based on counting the number of direct relations between concepts (i.e. the links or edges) in the graph path between two concepts. A well-known measure is the one proposed by Wu and Palmer [1994], which estimates the similarity between two concepts a_n, a_l by considering their respective depths, and the depth of the Lowest Common Subsumer (LCS) of both concepts, as follows:

$$sim(a_n, a_l) = \frac{2 \times depth(LCS)}{depth(a_n) + depth(a_l)} \tag{3.1}$$

Another one is the Leacock and Chodorow measure [1998], which estimates the similarity based on the distance (*d*) of the shortest path between the two concepts, and the maximum depth of the taxonomy (D):

$$sim(a_n, a_l) = \log \frac{2 \times D}{d} \tag{3.2}$$

While these approaches are intuitive, they are based on two assumptions that are rarely true in many domain hierarchies: (1) concepts and relations are uniformly distributed, and (2) relations at the same level in the taxonomy correspond to the same semantic distance between concepts (i.e. all the links have the same weight).

Node-based measures rely on comparing the properties of the concepts involved, which can be related to the concepts themselves, their ancestors, or their descendants. Specifically, these measures compare the Information Content (IC) of a concept, which is an estimation of how specific and informative a concept is. The IC of a concept is usually estimated as the negative log likelihood:

$$IC(c) = -\log p(c) \tag{3.3}$$

where the probability of occurrence $p(c)$ in a specific knowledge base is based on its frequency of annotation (i.e. how many instances/entities are annotated by the concept). Once estimated the IC of the domain concepts, it can be used to quantify the information that the common ancestors share and thus measure their semantic similarity.

Resnik [1995] proposed a node-based measure that calculates the similarity between two concepts by simply considering the (IC) of the LCS:

$$sim(a_n, a_l) = IC(LCS) \tag{3.4}$$

Jiang and Conrath [1997] proposed another measure using IC that, in addition to the IC of the LCS, the IC of the concepts themselves is also considered as follows:

$$sim(a_n, a_l) = \frac{1}{IC(a_n) + IC(a_l) - 2 \times IC(LCS)} \tag{3.5}$$

Analogously, Lin [1998] proposed another variant which augments the information content of the LCS with the sum of the IC of both concepts:

$$sim(a_n, a_l) = \frac{2 \times IC(LCS)}{IC(a_n) + IC(a_l)} \tag{3.6}$$

The advantage of node-based measures is that they are less sensitive to the issues of variable semantic distance and variable node density than link-based measures, since the IC gives a measure of a concept's specificity that is independent of its depth in the ontology's hierarchy. However, the IC may be biased by the popularity of some concepts, i.e., concepts that are more frequently annotated than other concepts.

Hybrid (node- and link-based) measures use a combination of the previous two types in order to overcome their individual drawbacks and obtain more accurate similarity estimations. An example is the work of Li et al. [2003], which proposed a link-based measure making use of path length and hierarchical depth as well as node density structural information. The hybridization of the approach is based on how they calculate the node density, which is measured using the IC of the concept. Then, these factors are combined in a non-linear way.

More recently, Passant [2010] proposed a set of link-based similarity measures, called Linked Data Semantic Distance (LDSD), which exploits any kind of semantic relationship and not only hierarchical ones as the previous approaches. The author describes a particular scenario for using *DBpedia*.

3.2.2. *Distributional Measures*

Distributional similarity measures only require a data corpus (i.e. text documents or a set of annotated items) to derive similarity assessments between concept pairs; they are based on statistics about the distributional properties of the concepts. The fundamental idea behind the use of distributional information to extract semantic similarities is the so-called distributional hypothesis of meaning**:**

Distributional hypothesis: concepts repeatedly co-occurring in the same context tend to be related.

In *linguistics,* several formulations to this effect can be found, such as [Firth 1957]: “You shall know a word by the company it keeps”, [Rubenstein and Goodenough 1965]: “words which are similar in meaning occur in similar contexts”, and [Schütze and Pedersen 1995] “words with similar meanings will occur with similar neighbors if enough text material is available”. When using raw text as data corpus, here “context” refers to other words that occur within a certain window (e.g. a sentence or a paragraph), and the intuition is that semantically related words are likely to be used in text in a similar way and thus they tend to co-occur more often.

In this thesis, we claim that this formulation of distributional hypothesis can be easily generalized to other types of data like for instance items’ descriptions in the form of attribute-based profiles, which is the type of data available in CB recommenders. For example, in the movie domain we could assess that the movie actor *Bruce Willis* is somehow semantically related to *action* movies, if these two concepts repeatedly co-occur in the same movies’ profiles.

A popular representation method to measure distributional relatedness between concepts is to use a vector space representation of concept meaning (the VSM) and measure the similarity in terms of proximity in such vector space. When using item content data in a CB recommendation application, the concepts’ meaning (also known as semantic vectors) is represented in terms of the items in which they occur; therefore, a *concept-by-item* co-occurrence matrix is commonly built where each row corresponds to a concept’s semantic vector. According to the results of our empirical evaluation presented in Chapter 4 - a better alternative to represent the attributes’ meaning in CB recommendation is with respect to the user profiles, since in this way the semantic relatedness also considers the user’s interests. Based on how they build the co-occurrence matrix and estimate the semantic similarities between concepts three main types of distributional measures can be identified: set theory measures, probabilistic measures, and dimensionality reduction techniques in combination with the cosine similarity.

Set theory measures employ the Boolean representation model, where true indicates that the concept appears in the item and false otherwise. Therefore, the frequency-based weights associated to the concepts, like TF-IDF weights, are not considered. A popular set theory measure is the Jaccard index, which computes the similarity between two concepts a and a^* based on the intersection of their binary semantic vectors (w_a and w_{a^*} respectively), divided by the size of their union:

$$sim(a, a^*) = \frac{w_a \cap w_{a^*}}{w_a \cup w_{a^*}} \tag{3.7}$$

Probabilistic information-theory measures compare the semantic relatedness of two concepts based on the similarity of their probability distributions. Consequently, here semantic vectors are built by using a probabilistic weighting scheme, where each entry is computed as the conditional probability of a concept a co-occurring with another concept a^* :

$$w_{nl} = P(a^* \, | a \,) = \frac{P(a \cap a^* \,)}{P(a)} \tag{3.8}$$

Note that using this method the concept's meaning is represented in terms of the other concepts, i.e. a *concept-by-concept* co-occurrence matrix. Once computed the probabilistic semantic vectors, the similarity between two concepts can be estimated using a probabilistic similarity measure. A popular measure is the Kullback and Leibler [1951] distance, which estimates the concept-to-concept similarity by normalizing their relative entropy, as follows:

$$sim(a, a^*) = \frac{1}{1 + \sum_{i=0}^{l} \log \frac{w_{ai}}{w_{a^*i}} \times w_{ai}} \tag{3.9}$$

Dimensionality reduction techniques reduce the usual high dimensional and sparse representations of semantic vectors to a low dimensional and more compact one before estimating similarities between two concepts based on the cosine similarity. Commonly, when using these techniques the original semantic vectors are previously weighted using a frequency-based weighting scheme, like TF-IDF. Depending on the technique used to reduce the dimensionality of the semantic vectors, three main methods can be identified:

- Latent Semantic Analysis (LSA) [Deerwester et al. 1990], which is a popular method, historically in the IR field and more recently also in the RS field, to alleviate the limitations related to the exact string matching, like synonymy detection, and also to detect latent concept associations that reflect higher-order co-occurrences. This technique uses the conventional SVD to reduce the dimensionality of the co-occurrence matrix to a *l*-dimensional space; *l* being the number of factors that will describe a semantic vector w_n, and whose exact value is data-dependent and thus it has to be identified experimentally (see Section 2.4.3 for more details about SVD).
- Random Indexing (RI) [Kanerva et al. 2000], an alternative method to dimensionality reduction based on the Johnson and Lindenstrauss [1984] lemma, which claims that a vector space can be efficiently reduced by projecting the points into a randomly selected subspace of enough high dimensionality. This approach has the advantage of estimating the semantic vectors in an incremental way rather than requiring the entire co-occurrence matrix, as in LSA.

- Reduction based on human-based (explicit) domain concepts, where contexts of co-occurrence or usages are mapped to an explicit low dimensional set of concepts that categorize them. An instantiation of this method is Explicit Semantic Analysis (ESA) [Gabrilovich and Markovitch 2007], in which items are classified in terms of *Wikipedia*-based concepts and semantic vectors are represented as vectors of this set of concepts.

Once the lower-dimensional and more compact semantic vectors are computed, concept-to-concept similarities are estimated as the cosine of the angle between their vectors (w_a and w_{a^*}):

$$sim(a, a^*) = \frac{w_a^T w_{a^*}}{\sqrt{\sum_{i=0}^{l} w_{ai}^2} \times \sqrt{\sum_{i=0}^{l} w_{a^*i}^2}} \qquad (3.10)$$

3.2.3. *Limitations of Semantic Similarity Measures*

Ontology-based and distributional measures use distinct knowledge sources, each of which has its own drawbacks and advantages [Mohammad and Hirst 2012]:

- **Ontologies are more expensive than raw data.** Ontology engineering requires human experts and is usually a time consuming task; this limits its use in many domains and the number of publicly available domain-specific ontologies is limited. In contrast, in domains where raw data are easy to collect, the application of distributional measures is less expensive. Updating ontologies is also more costly than re-building distributional semantic representations, which only require re-training the semantics model with the new data.
- **Ontologies are static representations of a domain that may not correspond to the actual data**. Ontologies are fixed specifications of a domain based on the criteria of human experts, and the accuracy of ontology-based measures depends on the quality of the representation, which may not suit the data. In contrast, distributional measures are not bounded by a fixed semantic representation and can identify *as similar* concepts that are semantically related, but whose relation is not explicitly defined in any ontology.
- **Distributional measures are more affected by data sparsity**. A disadvantage of distributional measures is that they may assign low similarity values to clearly related concepts simply because there is not enough co-occurrence evidence of such relation in the data set. On the contrary, this is not an issue for ontology-based measures that rely on the structure of ontologies (i.e. *link-based* measures). However, *node-based* measures, which use the concept's IC, could be also affected by data sparsity.

3.3. Semantically-Enhanced Content-Based Filtering

This section reviews related work on SERSs that incorporate attribute-to-attribute ontology-based or distributional similarities into an existing CB recommendation technique, which can be the component of a hybrid recommender.

This semantic knowledge can be incorporated into several stages of the CB recommendation process: (1) in the item representation or content analysis phase (Section 3.3.1), (2) in the user-profile learning phase (Section 3.3.2), and (3) in the profile matching or prediction phase (Section 3.3.3), where the affinity between different concept-based profiles of items or users is estimated.

3.3.1. *Semantic Content Analysis*

The main goal of incorporating semantic knowledge into the content analysis process is to mitigate the lack of semantics of traditional keyword-based item representation. The basic idea consists of mapping the keyword-based profiles to a semantic concept-based representation, which then is commonly used to learn user profiles in the same semantic space and generate recommendations based on the affinity between user and item profiles. These methods perform a different kind of concept-based item representation depending on whether they use distributional or ontology-based measures.

SERS techniques using ontology-based measures aim at mapping keyword-based item profiles to explicit concepts defined in the ontology. Particularly, attribute-to-attribute similarities are exploited for word disambiguation and classification into concepts of the ontologies. Normally, SERS techniques that use this semantics exploitation method deal with text-based descriptions of items and employ thesauri-based ontologies like *WordNet*. For instance, *SiteIF* [Magnini and Strapparava 2001] is a news recommender that employs a node-based measure to map keywords to *WordNet* concepts (also known as "synsets"). The authors use, as ontology, *MultiWordNet*, a multilingual lexical database where English and Italian WordNets are aligned. *SEWeP* (Semantic Enhancement for Web Personalization) [Eirinaki et al. 2003] is a Web recommender that employs a link-based measure to map keywords to *WordNet* concepts and to concepts of a domain-specific hierarchy. *ITR* [Degemmis et al. 2007] is another example, in this case a general purpose recommender, which extracts keywords from text-based product descriptions and also exploits link-based similarities to map keywords to *WordNet* synsets.

In contrast, SERS techniques using distributional measures aim at mapping keyword-based item profiles to semantic latent concepts rather than explicit ontology concepts**.** Therefore, these methods use dimensionality reduction techniques, such as LSA and RI, to directly produce more informative and less sparse item profiles, but not to estimate attribute-to-attribute semantic similarities. For example, Mobasher et al. [2003] proposed a movie recommender that uses LSA to reduce the item profiles into a latent semantic space. Then, they compute item-to-item similarities based on this reduced representation using the cosine similarity and exploit them into a content-boosted item-based CF hybrid recommender.

Bambini et al. [2011] proposed *CB-LSA*, a TV-program recommender that also uses LSA to reduce item profiles to a latent semantic space. In this case, the authors obtained both items and user profiles represented in the same reduced vector space by means of a projection technique known as *folding-in*, and produced recommendations based on the cosine of the angle between the reduced latent profiles. Assuming that $A_k = U_k \times S_k \times V_k^T$ is the factorization of the *attribute-by-item* matrix, where k is the number of latent concepts describing items, and w_i is the keyword-based representation of item *i*, the folding-in projection is defined as follows:

$$\widetilde{w}_i = U_k^T w_i \tag{3.11}$$

Similarly, the representation of user *u* is also calculated in the same latent space as the projection of the vector of user ratings. Denoting as w_u the vector of user ratings over the set of items *I*, its projection is defined by:

$$\widetilde{w}_u = w_u V_k S_k \tag{3.12}$$

More recently, Musto et al. [2013] proposed a music recommender that also represents both item and user profiles in a latent semantic space. But instead of using LSA, they employ RI as dimensionality reduction technique. Additionally, they represent user interests by means of two vectors, one based on positive ratings and one based on negative feedback. Differently from the previous approach, they do not use the folding-in projection technique to learnt user profiles. Instead they estimate the user profiles as the weighted sum of item vectors in the latent space ($\widetilde{w}_i$) that the user has rated, as follows:

$$\widetilde{w}_{+u} = \sum_{i \in I_u^+} \widetilde{w}_i \times r_{ui} \tag{3.13}$$

$$\widetilde{w}_{-u} = \sum_{i \in I_u^-} \widetilde{w}_i \times r_{ui}$$

3.3.2. *Semantic Expansion of User Profiles*

CB recommendation techniques require a minimum amount of ratings per user in order to learn accurate user profiles (i.e. suffer from *new user* cold-start). To mitigate this sparsity-related limitation, one can exploit the semantic associations between attributes to expand the user profile learnt from the users' ratings by means of interest propagation methods. These propagation methods are known as Spreading Activation (SA) strategies and come from past work in *associative* IR [Crestani 1997], a specific form of IR in which relevant information is delivered to the user by retrieving information that is "associated" with some information the user already retrieved and considered as relevant.

The general process of a SA strategy is defined by a sequence of iterations each one consisting of: (1) one or more pulses; (2) termination check. Basically, a pulse is made of three phases: pre-adjustment, spreading, post-adjustment. The pre- and post-adjustment phases, which are optional, are used to control both activation of single nodes and the overall activation of the network. The spreading phase consists on a number of passages of activation weaves from one node to its linked nodes. Pulse after pulse, the activation weight spreads over the network reaching nodes that are far from the initially activated ones. After a determined number of pulses have been fired a termination condition is checked. If the condition is verified than the SA process stops, otherwise it continues for another series of pulses. The result of the SA process is the activation level of nodes reached at termination time.

Most SA strategies employ specific types of constraints to control the spreading phase. For this reason, they are also known as *Constrained Spreading Activation* (CSA) strategies. The most typical ones are: *distance* constraints, which stop or degrade the weight propagation at some specified distance from the original node; *fan-out* constraints, where nodes connected to many other nodes receive special treatment in the activation process in order to avoid large spreading of the activation weight on the network; and *rule-based* or selective activation, which controls the degree of propagation to the target node based on the type of links. This latter constraint is usually used along with hierarchy-based inferences, which can be seen as a special case of SA strategies. Two main types of propagation through explicit hierarchical relations can be identified: (1) *upwards* or *bottom-up* propagation, in which activation weights from sub-concepts are propagated to the parent concept; (2) *sidewards* propagation, where weights from activated sub-concepts are spread across the other sub-concepts of the same parent concept.

SERS techniques that semantically expand the users' interests commonly employ a CSA strategy over the explicit relationships defined in the ontology. For example, *Deep Map* [Fink and Kobsa 2002] is a tourist recommender that exploits a hierarchy of types of Points of Interest (POI) by combining *upwards* and *sidewards* propagation. Both types of propagation are constrained based on a threshold that delimits the minimum number of activated sub-concepts necessary to propagate the interest weight to the parent concept and/or sub-concepts. In addition, the authors used a decay factor of 50% when the weight is spread via *upwards* propagation, and of 70% in the *sidewards* case, since this type of propagation is less reliable according to the authors. *QuickStep* [Middleton et al. 2004] is a scientific paper recommender that exploits an ODP-based taxonomy of topics describing papers by means of *upwards* propagation of interest weights. In this case, the interest weight of a concept is always propagated to the parent with a decay factor of 50%. Additionally, as a type of time-aware contextualization, the authors also use a time decay factor in order to reduce the weight of old interests.

News@hand [Cantador et al. 2008; 2011] is a news recommender system that employs a more sophisticated CSA strategy using the IPTC *NewsCodes* classification[4]. The proposed CSA strategy exploits domain-specific relations besides hierarchical ones, whose weights are used as decay factor. The

[4] See [http://www.iptc.org/], accessed on November 14th, 2013.

authors use a manually assigned decay factor for each type of relation in the ontology, and also the following additional constrains:

- a threshold delimiting the minimum interest weight necessary to propagate it,
- a threshold delimiting the maximum number of propagation steps (i.e. pulses),
- and a threshold controlling the maximum fan-out in order to reduce the “hub effect” in concepts with many relations to other concepts;

Following the *interactional view* of context (see Section 2.5.1), besides using the CSA strategy to expand long-term user profiles, the authors also use it to expand short-term user profiles that contain user’s interests shown in the current session. Once they have computed both profiles, a “contextualized” user profile is learnt as the intersection of both the long- and short-term user profiles. Finally, they make recommendations based on the affinity between the expanded user profiles and the item profiles estimated as their *cosine* similarity.

Sieg et al. [2010] proposed a book recommender system that exploits a taxonomy derived from the Amazon.com’s book classification by using a simplified CSA strategy compared to the previous work. The main difference is the use of a link weighting mechanism that assigns a weight to each concept-to-concept association by means of an ad-hoc *node-based* similarity measure. In this case, they use the semantically expanded user profiles to enhance a user-based CF meta-level hybrid by computing more accurate user-to-user similarities; the authors use the Euclidean distance to compute similarities between user profiles.

The *CHIP Art Recommender*[5] [Wang et. al 2010] is an artwork recommender that makes use of a domain-specific ontology describing artworks, which is adapted from different ontologies such as Getty vocabularies[6] (ULAN, AAT, TGN) and Iconclass thesaurus[7]. The main novelty of this approach with respect to the others is that, apart from ontology-based semantic associations that have manually-assigned weights, they use weighted associations derived by using distributional semantics; specifically, they use the corrected Jaccard measure. When a concept is activated from both ontology-based and Jaccard similarities, then a linear combination of the weight is propagated from each source node. Finally, the system recommends the top-n artworks whose attributes better match the expanded user profile.

Besides expanding user preferences, some SERSs employ CSA strategies for directly making recommendations. For example, *InterestMap* [Liu and Maes 2005] is a recommender system that suggests topics of interest to users of a social network by means of a CSA strategy. The authors use a probabilistic

[5] See [http://chip.win.tue.nl/home.html], accessed on November 14th, 2013.

[6] See [http://www.getty.edu/research/tools/vocabularies/index.html], accessed on November 14th, 2013.

[7] See [http://www.iconclass.nl/], accessed on November 14th, 2013.

distributional measure over the corpus of user profiles in order to weight the semantic associations between topics. The user profiles were crawled from a social network website and then normalized to an explicit set of topics defined in different domain ontologies. The distributional similarities between topics are represented as a dense semantic network, over which the CSA strategy is applied to suggest new topics highly related to the ones the user is interested in. The strategy uses two types of constrains: a threshold that delimits the minimum weight of a topic-to-topic association to be considered, and a decay factor of 25% per level of distance from the source nodes. Another example is *Avatar* [Blanco-Fernandez et al. 2008], a TV program recommender that exploits a domain-specific ontology, adapted from the standard TV-Anytime. The main novelty of their approach is that the weights of explicit relations are automatically computed by considering the user profiles in combination with an ad-hoc link-based similarity measure; therefore, semantic associations are also personalized to the each user. They make recommendations to each user by applying the SA strategy over the personalized semantic network (i.e. here user profiles are both the attributes and their weighted relations), and finally, the TV programs that contain attributes with higher activation values are recommended.

More recently, Kaminskas et al. [2012] proposed a location-aware music recommender that provides recommendation of musicians contextualized according to the POI the user is currently visiting. In order to link musicians and POIs, the authors use the *DBPedia* ontology that connects items from different domains through different kinds of domain-specific relations. Once the current POI of the target user is identified, they apply an adaptation of the SA strategy proposed in Cantador et al. [2008; 2011] to recommend musicians highly related to the target POI.

3.3.3. ***Semantic Profile Matching***

Here we present SERS techniques that only exploit attribute-to-attribute semantic associations in the prediction phase. Rather than semantically expand user profiles by propagating the initial set of interests, here the goal is to produce better recommendations by directly including the semantic similarities into the prediction function.

SERSs of previous subsections commonly use, after applying a SA strategy, traditional vector-based similarity measures to estimate the affinity between two concept-based profiles (i.e. user-to-item or user-to-user or item-to-item profile matching). The limitation of these measures is that they only use syntactic matching strategies (i.e. syntactically different concepts do not contribute to the similarity value), and thus they have a lack of semantics intelligence in this sense. *Pairwise* profile-to-profile matching strategies are a solution to this limitation, because they compare two profiles using a specific aggregation of the individual pairwise concept-to-concept similarities, allowing for a finer-grained comparison than syntactic matching. Depending on how this aggregation is produced, pairwise measures can be classified in two main variants: *all-pairs*, which consider every pairwise combination of concepts from the two sets; and *best-pairs*, which only consider the best-matching pair for each concept of one of the sets. In these measures a global similarity score between two entities is obtained by aggregating the considered concept-to-concept similarities. The aggregation method is typically the average, although other methods such as the maximum, the minimum and the sum can also be used.

Ganesan et al. [2003] proposed several pairwise strategies exploiting hierarchy-based similarities, including:

- an *all-pairs* strategy that generalizes the *cosine* similarity;
- a *best-pairs* strategy, called *Optimistic Genealogy Measure*, that employs a tree-like hierarchy to compute the similarity of two sets.

In particular, the authors use the Wu and Palmer [1994] *link-based* similarity to estimate attribute-to-attribute similarities. Although they do not evaluate the effectiveness of the measures in a recommendation task, they claim that the proposed measures can be useful to estimate more precise user-to-user similarities in a user-based CF approach.

Liu et al. [2007] proposed a product recommender that exploits a taxonomy of user requirements by means of a pairwise strategy. Concretely, they use an *all-pairs* measure to estimate *user-to-user* similarities, which sums up all the pairwise similarities between concepts of both user profiles that have a positive semantic similarity value. The global similarity is then normalized dividing the score by the size of the smallest profile. The concept-to-concept semantic associations are estimated using an ad-hoc *link-based* measure weighted by hierarchical depth.

Shoval et al. [2008] proposed *ePaper*, a news recommender that exploits a three-level taxonomy, extracted from the IPTC classification, by means of a pairwise matching strategy. Specifically, this approach uses a *best-pair* strategy to estimate the affinity between a concept-based user profile and an item profile, in which for each attribute of the item profile only the best matching concept of the user profile is considered. They compute a weighted sum according to the weight of the hierarchical relation. The weights of the relations are manually assigned depending on the type of relation between two concepts in the hierarchy (e.g. a weight of 2/3 is used when the item concept is a sub-concept of the user's concept). Finally, they obtain the global similarity dividing the weighted sum over the item's concepts by the sum of weights of the user's concepts.

3.4. Case Study: Exploiting a Movie Taxonomy in Netflix Data

This section presents research on ontology-based SERSs, which consisted of evaluating an existing approach using the Netflix data set. The technique described here was original implemented in a tourism recommender system, and comes from my Master thesis in AI [2009; 2012]. Its main novelty with respect to the research described in previous section is that exploits hierarchy-based relations using a combination of two ontology-based semantics exploitation methods: (1) a CSA user-profile expansion, (2) and a pairwise profile matching strategy. This experimental evaluation on the Netflix data set was motivated by the fact that previous experimental evaluations on ontology-based SERSs were difficult to reproduce, since they were performed on specific application domains whose data sets were not publicly available. Furthermore, in most evaluations the performance was not compared to state-of-the-art CF recommendation techniques.

In the following subsections we briefly describe the evaluated ontology-based SERS technique (Section 3.4.1), the characteristics of the data and taxonomy used in the evaluation (Section 3.4.2), and the experimental evaluation and results (Section 3.4.3).

3.4.1. *Ontology-Based Content-Based Filtering*

The ontology-based SERS technique proposed by Codina [2009; 2012] extends a traditional CB recommendation technique by incorporating ontology-based semantics in two stages of the CB recommendation process: (1) after the user-profile learning method using a CSA strategy, and (2) in the item-to-user profile matching using a pairwise strategy.

User-profile expansion. Similarly to Fink and Kobsa [2002], the proposed CSA strategy combines *upwards* and *sidewards* propagation to semantically expand user profiles, which are learnt from the training rating data using an ad-hoc *rating average* method. The spreading is constrained based on a threshold that delimits the minimum percentage of activated sub-concepts necessary to propagate the interest weight to the parent concept and/or their sibling concepts. These thresholds have to be experimentally optimized for each type of hierarchical propagation (see Section 3.4.3 for the exact values used on the Netflix data set). The main novelty with respect to the CSA strategy proposed by Fink and Kobsa [2002] is that, after an *upwards* propagation, the weight of a parent concept is always updated; no matter if the concept already exists in the user profile.

Figure 3.1 shows the pseudo-code of the CSA strategy that expands the profile of a target user u. The input of the algorithm is the weighted interest profile w_u of the user, which has been learnt from her training ratings, the upwards and sidewards thresholds, and the maximum depth of the taxonomy.

```
Input
  $w_u$: initial user profile
  $t_{up}$: upwards propagation threshold
  $t_{sp}$: sidewards propagation threshold
  D: maximum depth of the taxonomy
Output
  $w'_u$: semantically-expanded user profile

  depth = D;
  while depth > 1 do
    conceptFamilyList = getFamiliesOfConcepts($w_u$,depth);
    foreach $f_u$ ∈ conceptFamilyList   do
      percentage = getPercentageActivedConcepts($f_u$);
      if percentage > $t_{up}$ then
        $w'_u$ = doUpwardsPropagation($f_u$);
      endif
      if percentage > $t_{sp}$ then
        $w'_u$ = doSidewardsPropagation($f_u$);
      endif
    endfor
    depth--;
  endwhile
return $w'_u$
```

Figure 3.1. Hierarchy-based Spreading Activation

Firstly, all the concepts of the user profile with weight different from zero are grouped by level of depth in the taxonomy. Then, starting for the deepest concepts (i.e. the most specific attributes), *upwards* and *sidewards* propagation are applied in the families of concepts, whose percentage of activated sub-concepts is larger than the corresponding thresholds. A concept is activated if its interest weight is different from zero. The parent concepts that are activated from the *upwards* propagations are included in the set of concepts of the next level of depth. If the parent concept already exists in the set, then the weight estimated trough the propagation is linearly combined with the interest estimation based on ratings. This spreading process is executed for each level until the root node is reached. The result is a semantically expanded user profile w'_u.

- The function *doUpwardsPropagation(f_u)* applies the *upward* propagation to the family of sub-concepts f_u, propagating the average interest weight to the parent concept. If the parent concept already has an interest weight, a weighted combination of the existing and propagated interest weight is used, based on a measure of how trustworthy the interest prediction is.
- The function *doSidewardsPropagation(f_u)* applies the *sideward* propagation to the family of sub-concepts f_u, propagating the average interest weight to the sibling concepts. In this case, the average weight is only propagated to the unknown concepts.

Pairwise item-to-user profile matching. The *item-to-user* profile matching strategy follows an *all-pairs* combination. Codina [2009; 2012] proposed a rating estimation function assuming that items are represented as binary vectors where $w_{ia} = 1$ indicates the item *i* contains the attribute a. In this case, the *all-pairs* function estimates the rating that a target user *u* will provide to an item *i* as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + \sum_{a \in w_i} \frac{\sum_{a^* \in w_u} w_{ua^*} \times sim(a, a^*)}{\sum_{a^* \in w_u} sim(a, a^*)} \tag{3.14}$$

The $sim(a, a^*)$ function returns the semantic similarity value between two attributes, which is calculated using an ad-hoc *link-based* measure proposed by the author that, differently from the ontology-based measures presented in Section 3.2.1, weights differently hierarchical relations based on the depth of the deepest concept. The basic assumption of the proposed measure is that two upper-level concepts are less similar than two lower-level ones. The weight of the links between a sub-concept and its ancestor is controlled by a decay factor *K*, whose value depends on the depth of the sub-concept; being smaller as deeper the sub-concept. The exact values of *K* per level have to be learnt experimentally (see Section 3.4.3). Particularly, denoting *d* as the distance between the two concepts in terms of the number of links between them (e.g. d=1 when the ancestor is the sub-concept's parent) this link-based measure estimates the similarity between a sub-concept a and its ancestor a^* by means of the following recursive function:

$$sim(a, a^*) = \begin{cases} Sim(d) = 1 - K, & if\ d = 1 \\ Sim(d) = Sim(d-1) - K \times Sim(d-1), & if\ d > 1 \end{cases} \tag{3.15}$$

3.4.2. *Netflix Data and Taxonomy*

The Netflix dataset used here consists of 480,000 users, 17,700 movies and a total of 100,480,507 user's ratings ranging between 1 and 5. The original data set does not include descriptive content information about movies, only the title and release year. For this reason, we first had to obtain this information from other sources. Because the data set also included an internal Netflix ID for each movie, we decided to extract the movie information through the Netflix API[8]. Compared to typical parsing methods based on string matching of movie title in which often only a percentage of movies is well parsed, using the Netflix API we were able to collect attributes for all the movies in the data set. Specifically, the movie attributes that we extracted were actors, directors, genres, producer, language and region.

[8] See [http://developer.netflix.com/], accessed on November 14th, 2013.

After movie annotation, we manually designed a movie taxonomy based on the publicly available classification on Netflix website[9]. The taxonomy we defined contains 787 concepts, which are associated to specific values of the attributes *genre*, *producer*, *language* and *region*. The maximum depth in the taxonomy is 4, as can be observed in **Figure 3.2**. In order to simplify the process of mapping movie attribute values to ontology concepts, we used the same labels as concept names. Therefore, every attribute value was directly associated to one concept of the taxonomy.

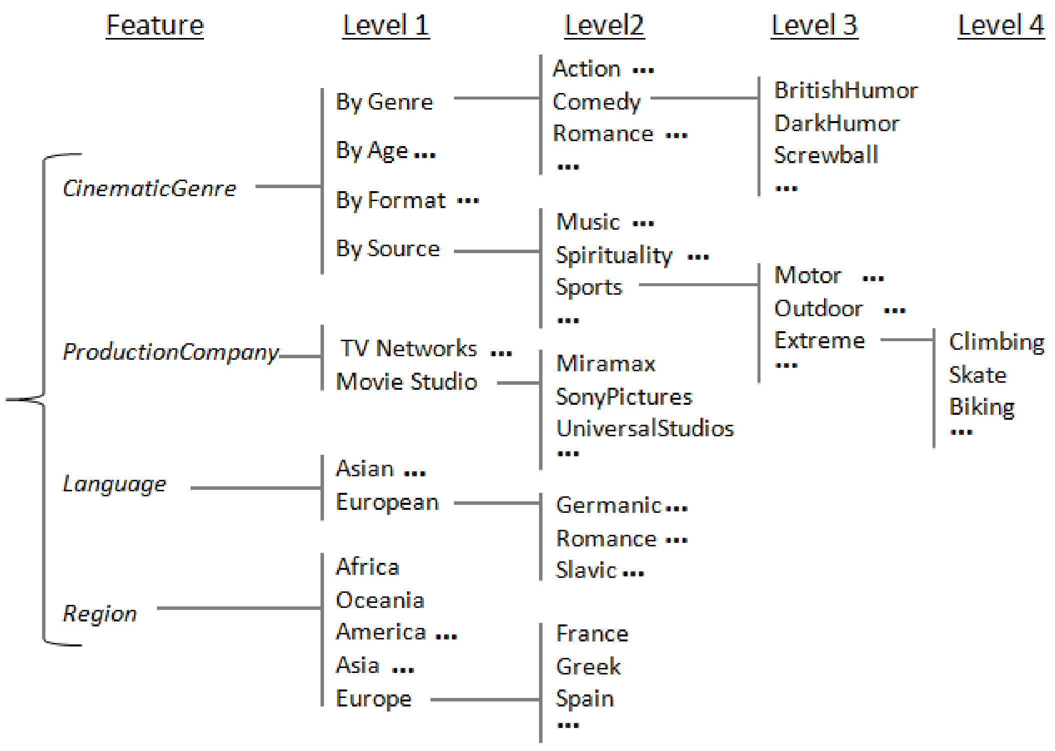


Figure 3.2 Partial representation of Netflix's movie taxonomy

3.4.3. *Experimental Evaluation and Results*

To maintain compatibility with results published by other researchers, we adopted some standards that were used for the Netflix contest, which consisted of measuring the prediction accuracy of the evaluated models in terms of Root Mean Squared Error (RMSE) over the ratings in the test set. We used for testing the *qualifying* set of ratings, which was provided by Netflix once the contest concluded. This set contains nearly 3 million ratings provided by users in the training set.

We optimized the meta-parameters of the ontology-based SERS technique using as validation the *probe* set, which is a subset of the training set provided by Netflix that contains 1.5 million ratings approximately. For the CSA strategy, we found the best performance when setting to 0.2 the *upwards*

[9] See [http://www.netflix.com/AllGenresList], accessed on November 14th, 2013.

threshold and to 0.85 the *sidewards threshold*. This means that, in this data set, the *upwards* propagations are clearly more useful than *sideward* ones. For the *link-based* similarity measure we set the following values for the decay factor depending on the level of depth: 0.3 when the deepest concept was at level four, 0.4 when it was at level three, and 0.5 when it was at level two.

Table 3.1 shows the prediction accuracy of the evaluated ontology-based SERS technique (*Sem-CB*) with respect to three baseline algorithms:

- *AVG*, which computes predictions by simply using the overall movie rating average.
- *CB*, which is uses a traditional item-to-user profile matching based on the dot product between item and user profiles (defined in Eq. 2.17), and hence without using the semantics exploitations methods. It also uses the same user-profile learning method based on rating average as *Sem-CB*.
- *Cinematch*, which was the Netflix's own algorithm and also used as the baseline algorithm of the Netflix contest.

In addition to the global prediction accuracy (i.e. measuring RMSE over all the test users), we also show the results over new users. In this experiment, we considered as new users the ones with 20 or fewer ratings. In terms of global RMSE, *Sem-CB* slightly improves the prediction accuracy with respect to *AVG* (2% gain) and *CB* (1% gain), and it is clearly worse than *Cinematch*. However, if we look at the cold-start performance, *Sem-CB* is clearly better than *CB* (6% gain) and *AVG* (7% gain), which demonstrates that *SEM-CB* is especially useful to enhance the performance of traditional *CB* techniques when users have few training ratings. Given that we did not have the exact implementation of *Cinematch*, we could not measure its RMSE for cold-start users.

Table 3.1. RMSE of evaluated models on Netflix movie rating data set. (*All* means that the result is averaged over all the users, and *New* averaged over the set of new users)

Models	RMSE	
	All	**New**
AVG	1.06	1.20
Cinematch	**0.95**	-
CB	1.05	1.18
Sem-CB	1.04	**1.12**

We also analyzed the effect of each semantics-exploitation method independently, in order to understand the usefulness of the proposed combination of strategies. From the analysis we concluded that the differences between a variant using only the CSA strategy and another one using the combined method (i.e. the CSA and the *all-pairs* strategy) were insignificant, indicating that in this case the combination did not help to improve the overall performance.

3.5. Semantically-Enhanced Context Modeling

This section reviews related work on CARSs, based on the representational view of context, which exploit the semantic relatedness between contextual situations during context modeling for improving context-aware recommendation. As we will show, most of these semantically-enhanced CARS techniques follow the contextual pre-filtering paradigm (see Section 2.5.2), since these context-aware approaches are especially affected by the data-sparsity problem.

Adomavicius et al. [2005] proposed *Generalized Pre-filtering*, a reduction-based variant that exploits the hierarchical relations between contextual conditions to determine the optimal aggregation of the used semantically-related situations (i.e. the optimal contextual segment). In this case, they do not measure the strength of the situation-to-situation similarity; they simply consider two situations as similar if at least one condition of each situation is hierarchically-related. The authors evaluated the method in a movie recommender that exploits a taxonomy describing three contextual factors (*time*, *place*, and *company*). This taxonomy contains few levels for each factor, particularly: three levels for *time*, two for *place* and four levels for *company*:

- Time: “Saturday” → “Weekend” → “Any time”
- Place: “Theater” → “Any place”
- Company: “Boyfriend” → “Friends” → “Not alone” → “Any company”

In this approach, the key process is the selection of the optimal contextual segments, which have to be identified experimentally by comparing their performance to a context-free recommendation approach. For example, the authors found that the segment [*Weekend*, *Theater*, *Any company*] improved significantly the accuracy of a user-based CF. Once selected the optimal segments, they build a local prediction model using the ratings tagged with situations that belong to each segment. Then, given a target contextual situation, the local model corresponding to the segment to which the target context belongs is used to make recommendations in that case. A limitation of this technique is that an exhaustive search for the optimal contextual segments can be infeasible if the number of possible combinations is too large.

Similarly to the previous approach, Liu et al. [2010] proposed a context-aware service recommender using a semantic pre-filtering approach in combination with user-based CF. But here the authors employ a more sophisticated ontology-based measure that exploits a context ontology based on Description Logics. Once computed the situation-to-situation similarities, they use them to weight the ratings that are used to make the neighborhood-based prediction.

Bouneffouf et al. [2012] proposed a mobile recommender that exploits hierarchy-based situation-to-situation similarities. The authors quantify the strength of the similarity by using a pairwise strategy, which calculates a similarity score as the weighted sum of condition-to-condition similarities of the same factor. To compute the similarity between conditions of a factor they employ the Wu and Palmer [1994] (*link-based*) measure. The weights of the sum correspond to the relevance of the contextual factor, which are manually assigned. The authors use the similarities between situations in an adaptation of the bandit algorithm that controls the degree of contextualization by means of balancing the exploration/exploitation trade-off.

Finally, Zheng et al. [2013b] proposed *Differential Contextual Weighting* (DCW), a pre-filtering approach which allows for a finer-grained context modeling compared to the DCR variant (proposed by the same authors and presented in Section 2.5.2) by exploiting situation-to-situation similarities. In this case they compute the similarity between two situations as the Jaccard similarity between their sets of known conditions. In particular, they use a weighted Jaccard measure in order to weight the contribution of each contextual factor according to its relevance. Similarly to DCR, the optimal weights are found by using the Particle Swarm Optimization method.

Chapter 4 - Exploiting Distributional Similarities for Enhanced Content-Based Recommendation

4.1. Introduction

The majority of current SERSs that aim at improving the effectiveness of CB recommendation techniques rely on semantic similarity measures using ontologies as main knowledge source, i.e., measures that estimate the semantic relatedness between two item's attributes based on their closeness in the domain ontologies. Taxonomies are the most abundant type of ontologies in recommendation domains, mainly because they are an inexpensive way to classify the available items of a catalog and can be easily re-used among systems in the same domain. The development of rich and expressive domain-specific ontologies is expensive, since it is a time consuming task which has to be performed by human experts. Typical examples of publicly available taxonomies used for improving CB recommendation techniques are *WordNet*, *Amazon*'s classifications, the IPTC *NewsCodes*, and the *Open Directory Project*.

The precision of ontology-based similarity measures strongly depends on how well the domain ontology represents the concepts and their relations. Hierarchies are built by grouping concepts based on their properties (e.g. shape, color, function, or role); and the order in which these properties are used to create the tree structure can result in dramatically different hierarchies, and thus different semantic similarity assessments between the same two concepts. For this reason, real-world taxonomies are able to capture certain semantic relations in their structure, while other relations are not represented. For example, publicly-available movie taxonomies often just describe movie genre classifications, but not relations between actors and genres or other combinations. This implies that only a small fraction of the potential semantic associations among domain concepts is exploited for recommendation.

A solution to this limitation, which does not require the use of domain-specific ontologies, consists of using distributional similarity measures rather than ontology-based ones. As explained in Section 3.2.2, distributional measures rely exclusively on a corpus of data to estimate similarities based on concept co-occurrence analysis. In addition to be based on a more-easily obtainable source of knowledge, distributional measures directly infer semantic associations from raw data, and hence they are able to infer data-specific relations that might not be defined in generic domain-ontologies.

This chapter presents our research on exploiting distributional similarities between items' attributes to enhance the accuracy of existing CB recommendation techniques. We have developed a novel SERS technique on top of a state-of-the-art CB approach, which we called *Semantic Content-Based* (SCB) filtering, making use of distributional measures that rely on both rating data and item content data. We present an extensive evaluation of SCB and compare their performance with respect to the state of the art, focusing on the comparison between distributional and ontology-based measures.

The remainder of this chapter is organized as follows. Section 4.2 positions our research with respect to the state of the art. Section 4.3 describes the proposed variants of SCB using different pairwise profile matching strategies. Section 4.4 presents the experimental evaluation of the SCB variants using a well-known movie rating data set. Finally, based on the experimental results, Section 4.5 highlights the main conclusions of this work.

4.2. Related Work

In the literature, the majority of the proposed SERS techniques for CB recommendation rely on ontology-based measures of semantic relatedness, and, depending on how these techniques exploit the semantic knowledge, two major approaches can be identified: based on Spreading Activation (SA) strategies (see Section 3.3.2) and based on *pairwise* profile matching strategies (see Section 3.3.3). The goal of SERSs using SA strategies is to expand the commonly sparse initial user profiles exploiting the semantic relations between attributes. Most of them produce recommendations by directly comparing the expanded user profiles and the profiles of the items to recommend by means of vector-based measures, such as the *cosine* and *Euclidean* measures [Cantador et al. 2008; Sieg et al. 2010]. Other works directly make recommendations after executing the SA strategy by selecting those items whose attributes better match the activated concepts [Liu and Maes 2005; Blanco-Fernandez et al. 2008]. In contrast, SERSs using *pairwise* strategies are less common in the literature. These approaches try to improve the concept profile matching by incorporating the semantic associations in the prediction phase [Liu et al. 2007; Shoval et al. 2008].

The majority of SERS techniques that use distributional measures are based on dimensionality reduction techniques, such as LSA [Mobasher et al. 2003; Bambini et al. 2011] and RI [Musto et al. 2013], and employ them to produce reduced and more informative item profiles. However, in all these approaches, which are explained in more detail in Section 3.3.1, the semantic associations between items' attributes are not exploited by means of spreading activation or pairwise matching strategies. Instead recommendations are produced by measuring the similarity between the reduced latent item and user profiles, typically using vector-based similarity measures.

Differently from these approaches, we propose a novel SERS technique that incorporates distributional similarities of items' attributes in the recommendation phase. In particular, we have developed and evaluated two variants of pairwise strategies using several distributional measures, comparing their performance to state-of-the-art SERS techniques based on the enrichment of user and item profiles. We also present a performance comparison in terms of prediction accuracy between distributional and ontology-based measures. Furthermore, in combination with the pairwise strategies, we have proposed the use of distributional similarities derived from co-occurrence analysis over the set of user profiles, proving that it is more useful to improve the effectiveness of recommendations than using item-based co-occurrences.

4.3. Semantic Content-Based Filtering

A requirement of SERS techniques using profile expansion or pairwise profile matching strategies is that both users and items must be represented using explicit concept-based profiles in the semantic space, where each concept correspond to a particular item attribute. Therefore, linear models (see Section 2.3.4), which learn weighted user profiles from the attributes of the items the user has rated, are the most suitable CB recommendation techniques to be extended with the proposed semantics-exploitation methods.

Here we present *Semantic Content-Based* (SCB) filtering, a prediction model which extends a linear CB technique by incorporating into the prediction model a pairwise profile matching strategy. **Figure 4.1** shows how the *pairwise* profile matching strategy is incorporated into the general process of a linear CB prediction model. As it can be observed, the pairwise matching strategy substitutes the direct vector-based profile matching, which is commonly used to compute the predicted rating for a target item and user, with a more sophisticated matching strategy that considers the attribute-to-attribute distributional similarities.

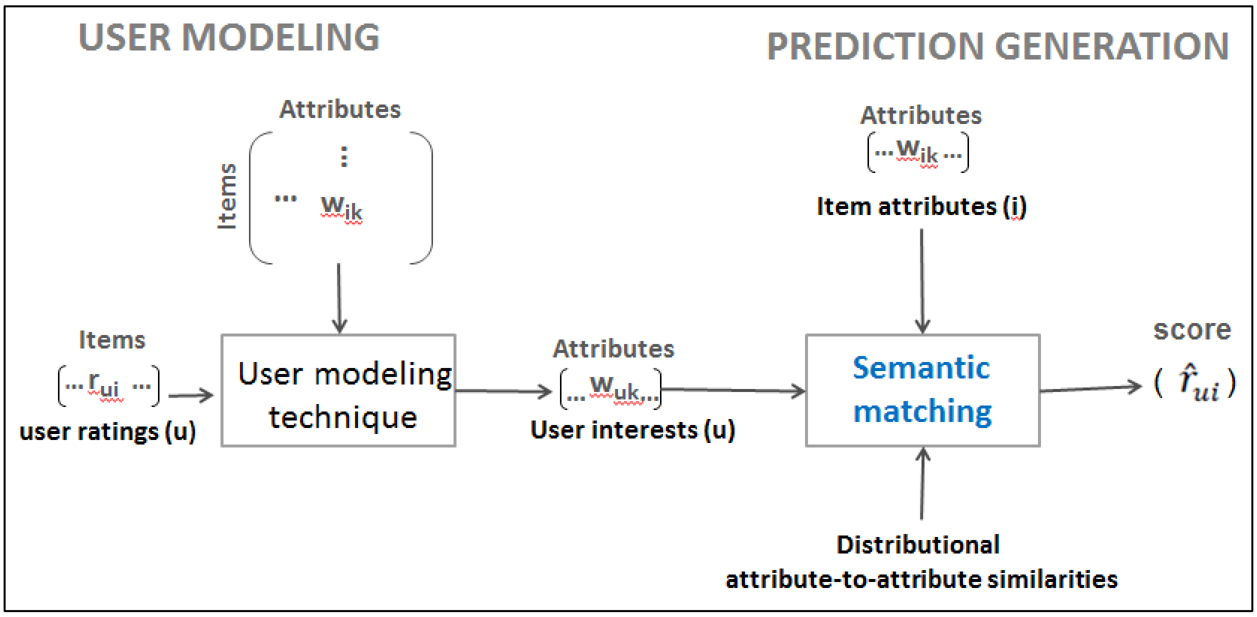


Figure 4.1. Semantic Content-based Filtering recommendation process

4.3.1. *User Profile Learning*

A key component of linear CB models is the user-profile learning method, since the predicted rating is estimated comparing the user's interests and the item's attributes, and in principle, the more accurate the user's profile, the better the predicted ratings. As it is difficult to decide at design stage which profile learning method is better for a given recommendation task and rating data, we have experimented with two *rating average* methods inspired from the state of the art.

One of these methods consist of an adaptation of the *rating average* proposed by Sen et al. [2009] and defined in Eq. 2.16, in which we normalize the user's ratings by subtracting from them the user and item biases (b_{ui}) during average computation (see Section 2.2 for details about baseline predictors). Given a user u and attribute a the interest weight of that user in that attribute (w_{ua}) is computed as follows:

$$w_{ua} = \frac{\sum_{i \in I_{ua}} (r_{ui} - b_{ui}) w_{ia}}{\sum_{i \in I_{ua}} w_{ia}} \tag{4.1}$$

Where I_{ua} is the set of items the user has rated which contain the attribute a in their profile, and w_{ia} is the weight associated to attribute a in the item profile, which can be a Boolean value or have a more sophisticated representation if TF-IDF is used.

Similarly to SVD++, proposed by Koren [2009] and defined in Eq. 2.26, which incorporates the actions of user's ratings as additional information for a MF prediction model, we have extended the previous rating average variant in Eq. 4.1 by including the action of rating items as additional implicit feedback. We use this additional information to learn an independent interest weight for a target user u and attribute a as the average weight of the item's attribute over the items the user has rated (I_{ua}):

$$w_{ua} = \operatorname{logit} \frac{\sum_{i \in I_{ua}} w_{ia}}{|I_{ua}|} - \operatorname{logit} \frac{\sum_{i \in I_{ua}} w_{ia}}{|I|} \tag{4.2}$$

$$\operatorname{logit}(p) = \log(\frac{p}{1-p})$$

As in the previous variant, we adjust the implicit interest weight by subtracting from it the overall attribute popularity (right term of Eq. 4.2). The log odds ratio here is used to normalize the averages to the same scale before subtracting. Finally, the extended rating average method consists of a linear combination of the previous two equations according to a factor $\in (0,1)$:

$$w_{ua} = \frac{\sum_{i \in I_{ua}} (r_{ui} - b_{ui}) w_{ia}}{\sum_{i \in I_{ua}} w_{ia}} \times \beta + \left[\operatorname{logit} \frac{\sum_{i \in I_{ua}} w_{ia}}{|I_{ua}|} - \operatorname{logit} \frac{\sum_{i \in I_{ua}} w_{ia}}{|I|} \right] \times (1 - \beta) \tag{4.3}$$

The purpose of this hybridization is to improve the precision of the interest weights. We experimentally found that the best accuracy results were obtained when using $\beta = 0.5$. For example, when using Eq. 4.1 alone, an attribute that appears only in one item rated by the user with 5 stars would have the maximum interest weight regardless the attribute's relevance; in contrast, using the combined method (Eq. 4.3), the weight could be reduced if the relevance of the attribute in the rated item (i.e. w_{ia}) is lower than average. Once combined the interest weights, we normalize them to the range [-1, 1] in order for all user profiles to have the same scale of degree of interest.

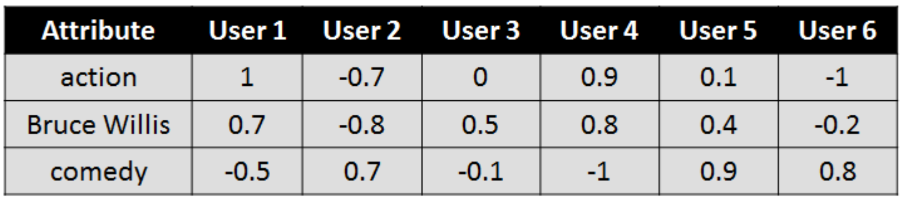
4.3.2. *User-Dependent Distributional Semantics*

As explained in Section 3.2.2, distributional similarity measures use a vector-space semantic representation of each attribute (known as semantic vectors) which represent the attribute's semantics based on its occurrence in the data corpus. The set of attribute's semantic vectors is known as the co-occurrence matrix, which is used to estimate the attribute-to-attribute similarities by comparing their respective semantic vectors.

Typically, SERS techniques using distributional measures rely on the co-occurrence analysis of the attributes in terms of how they are used to annotate the items of the domain, i.e., they measure distributional similarities based on the attribute-by-item co-occurrence matrix. However, we claim that *user-dependent distributional similarities* are more adequate than item-based co-occurrences to improve the accuracy of recommendations, since they do incorporate user's ratings in the calculation. Therefore, we investigate distributional measures that estimate semantic similarities between attributes based on how users are interested in them.

To measure *user-dependent distributional similarities*, first we need to build an attribute-by-user co-occurrence matrix where each entry stores a user interest weight, i.e. the attribute's semantic vectors are built with respect to the interest weights of user profiles. Then, we can assess two attributes as semantically related if several users are interested in them in a similar way (i.e. with a similar interest weight). Note that, in this case, the reliability of user-based semantic similarities depends on both the distributional measure used and the user-profile learning method employed.

Figure 4.2 illustrates as example the semantic vectors of three movie attributes with respect to six users of the system. If we analyze the number of co-occurrences between pairs of attributes, it is easy to observe that the semantic vectors of the <*Bruce Willis*, *action*> pair are more similar than the semantic vectors of the <*Bruce Willis*, *comedy*> pair.



Attribute	User 1	User 2	User 3	User 4	User 5	User 6
action	1	-0.7	0	0.9	0.1	-1
Bruce Willis	0.7	-0.8	0.5	0.8	0.4	-0.2
comedy	-0.5	0.7	-0.1	-1	0.9	0.8

Figure 4.2. Semantic vectors of three attributes with respect to six users

Based on this semantic representation, we calculate the distributional similarity between two attributes as the cosine of the angle between their semantic vectors. We chose the *cosine* similarity because it has been proved to be a reliable measure when dealing with high-dimensional vector spaces, and differently from set-theory and probabilistic measures, it also takes into account the value of the interest weights.

When the generated semantic vectors are sparse, the *cosine* similarity might cause misleading similarity estimations. In those conditions, we propose to apply dimensionality reduction techniques to produce more compact and informative semantic vectors. In addition to LSA, which uses the conventional SVD method to reduce the dimensionality of the co-occurrence matrix (see Section 2.4.3 and 3.2.2 for more details), we have decided to include in our study a novel method to produce latent semantic vectors of item's attributes based on the user's ratings proposed by Forbes and Zhu [2011], which uses a content-boosted MF recommender which extends the standard MF model by adding the item's attributes in the factorization process (see Section 2.6.1). In this case, the latent semantic vectors are a result of the joint matrix factorization process, and thus the number of latent factors is the same number as used for learning the latent vectors of user and items.

4.3.3. *Pairwise User-to-Item Profile Matching*

Although pairwise profile matching strategies are not commonly used by ontology-based SERS techniques, we want to demonstrate that they are the best way to exploit distributional semantics for CB recommendation. Here we propose two pairwise strategies that extend a linear CB prediction model, which estimates the rating for a given user and item as the dot product of their respective vector representations. In particular, we adapted the prediction model based on dot product defined in Eq. 2.17 by adding a normalization factor α that modifies the contribution of the profile matching based on the number of attributes that appear in both target profiles (in our experiments we obtained the best results when using $\alpha = 0.25$). Denoting M_{ui} as the set of attributes overlapping in both profiles (i.e. every attribute a_k whose w_{uk} and w_{ik} are different from zero), we predict the rating that a target user u will provide to an item i, as follows:

$$\hat{r}_{ui} = \mu + b_u + b_i + \frac{1}{|M_{ui}|^{\alpha}} w_i^T w_u \tag{4.4}$$

The proposed pairwise strategies extend the above mentioned prediction model by incorporating the attribute-to-attribute distributional similarities during item-to-user profile matching instead of simply using the dot product between profiles. We have proposed the following two pairwise strategies:

- An *all-pairs* strategy, where each item's attribute weight different from zero (w_{ia}) is aggregated with each user's interest weight (w_{ua^*}):

$$\hat{r}_{ui} = \mu + b_u + b_i + \frac{1}{|M_{ui}|^{\alpha}} \sum_{a \in w_i} \sum_{a^* \in S_{au}} aggregation(w_{ia}, w_{ua^*}) \times sim(a, a^*) \tag{4.5}$$

- A *best-pairs* strategy, in which each item's attribute weight different from zero (w_{ia}) is aggregated with the weight of the most similar user's attribute:

$$\hat{r}_{ui} = \mu + b_u + b_i + \frac{1}{|M_{ui}|^{\alpha}} \sum_{a \in w_i} aggregation\left(w_{ia}, \max_{a^* \in S_{au}} w_{ua^*}\right) \times sim(a, a^*) \quad (4.6)$$

S_{au} stands for the set of attributes in the user profile w_u whose semantic similarity to item's attribute a is larger than a *similarity* threshold. We used this threshold to avoid aggregating pairwise comparisons between item's and user's attributes weakly related, which may cause the introduction of too much noise to the predicted score. The $sim(a, a^*)$ function returns the similarity value between the two attributes, and the *aggregation* function defines how the two weights are aggregated. In addition to the typical product-based aggregation, we have experimented with the following aggregation functions: the maximum, the minimum and the average of the attribute weights.

Figure 4.3 illustrates the main differences between the *all-pairs* and *best-pairs* strategies. The example shows the pairwise combinations that are used in each strategy for the same item and user profiles (green lines represent the most similar pair for a given item's attribute, and the red ones the other possible combinations). To simplify, here we assume that the similarity between all the possible attribute pairs is larger than the *similarity threshold*. As it can be observed, the *all-pairs* strategy considers all the possible combination pairs and the *best-pairs* only the best-matching ones.

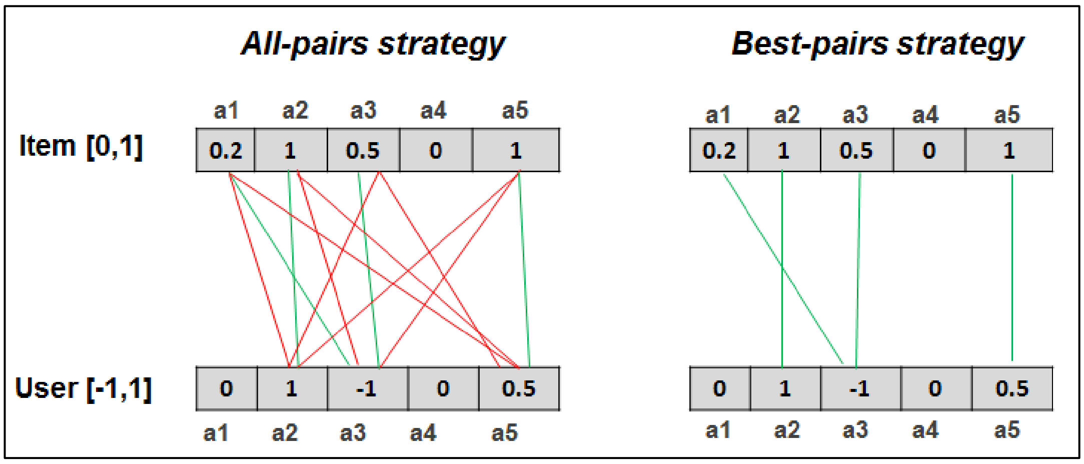


Figure 4.3. Proposed pairwise item-to-user profile matching strategies

In this work, we want to demonstrate that the *all-pairs* strategy is a better option when the predicted score is not bounded to a specific scale, like in ranking prediction, where what matters most is the order of the recommended items and not how similar the predicted and true ratings are. In contrast, in rating prediction, where the predicted scores must be normalized to the same scale of the real user's ratings, this additional aggregation of the *all-pairs* strategy may produce more "extreme" rating predictions (i.e. many 1-stars or 5-star ratings) and thus bigger errors than the *best-pairs*, which is more selective.

4.4. Experimental Evaluation

For the evaluation of SCB we used an extension of the well-known MovieLens[10] movie data set. This extended data set was released at the second International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011)[11], and includes the following movie attributes: *title, year of release, genres, directors, actors, countries of origin, filming locations* and *user tags*; some of them extracted from IMDb[12]. The data set consists of approximately 10 million ratings from 2113 users on 10.197 movies.

As in other machine learning techniques modeling explicit domain attributes, feature selection is crucial for the effectiveness of the model. Here we discarded the non-informative attributes, in order to maximize the effectiveness of the CB prediction models considered in the evaluation. Concretely, we experimentally found the best performance not considering the *countries* and *filming locations* as well as the least popular *actors* and *user tags*. In the case of actors, we only maintained the actors appearing in more than three movies and in the top-5 ranking of actors of at least one movie. In the case of tags, we removed all tags not associated to at least five different users and movies. Once selected the relevant tags, we then removed from the dataset all movies with less than five tags. **Table 4.1** shows the main statistics of the original movie rating data set and the pruned data set.

Table 4.1. Dataset statistics before and after pruning

Characteristic	Original dataset	Pruned dataset
Movies	10197	1646
Attributes	13367	3105
Ratings per user	404	239
Rating density	4%	14%

Taking advantage of the rating timestamps available in the data set, we have followed a per-user *all but n* evaluation protocol, in which for each user the *n* last ratings are selected as test set. As suggested by Shani and Gunawardana [2011], this evaluation protocol simulates better the online user behavior than traditional cross-validation protocols, and additionally, it is more suitable for significance testing because it allows obtaining more independent experiments to compare the algorithms (exactly one average per test user), assuming that test users are drawn independently from some population. For measuring the rating prediction accuracy of the considered models we computed the RMSE and the *Mean Absolute Error* (MAE) over the test set. Compared to MAE, RMSE penalizes more large errors.

---

[10] See [http://www.movielens.org], accessed on November 14th, 2013.

[11] See [http://ir.ii.uam.es/hetrec2011], accessed on November 14th, 2013.

[12] See [http://www.imdb.com], accessed on November 14th, 2013.

Besides the rating prediction accuracy, we have also measured the ranking performance of the evaluated models with the goal of validating the expected advantages of the *all-pairs* strategy for this recommendation task. In this case, we have followed a different evaluation protocol known as *one plus random* method [Cremonesi et al. 2010]. The method consists of selecting, for each user, a set of highly relevant items, and then the recommender generates a ranking for each of the selected relevant items plus the rest of candidate items, which are randomly selected. Finally, we measured the ranking precision of each ranking separately based on the final position of the relevant item in the ranking: the higher, the better.

Typically, in rating data sets the selection of highly relevant items is done by simply choosing the items rated with 5 stars. In our experiment, we use a finer-grained selection of test ratings using a user-specific threshold for determining the relevant items. The threshold is computed based on the particular user's rating behavior (i.e. a 4-star rating can be considered as relevant for a critic user, but irrelevant for a user rating many items with 5 stars). Particularly, we selected the last 5 ratings (according to their time-stamps) whose value is larger than the user-specific threshold. Regarding the selection of non-relevant candidate items, we randomly selected a certain number of items from the test set not rated by the user. In order to reduce the computation time of the experiments, we used a fixed number of candidate items per ranking. Experimentally, we found that using 750 non-relevant items was a good compromise between computational cost and reliability of the results.

For measuring the ranking precision we used the following two well-known metrics:

- *Recall*, which is defined as the number of "relevant" items (i.e. that appear in a higher position than a specific ranking cutoff) divided by size of the recommendation list. Because we used the *one plus random* protocol, we measured the recall for a given target item and user, and thus when we have a hit (i.e. if the target item appears in the top-n list) recall is equal to 1; and equal to 0 otherwise.
- *Normalized Discounted Cumulative Gain* (NDCG) [Järvelin and Kekäläinen 2002], which, differently from *Recall*, also takes into account the position of the item in the top-n list, i.e., a recommender that places the item in the first position is better than one placing it in the second position. The penalization is based on a relevance reduction logarithmically proportional to the position of the item.

All the reported results are averages of per-user evaluations, and the statistical significance of the differences between the proposed methods and the baseline algorithms have been calculated by means of the paired Wilcoxon sign rank test. In the case of *Recall* and NDCG the per-user evaluations are also averaged with the results at different ranking cutoffs, specifically from top-5 to top-50. Given that previous research on SERSs have demonstrated that these techniques usually perform especially well in cold-start scenarios, we have separately measured the performance of the models on users with few ratings. Concretely, we considered the 10% of users with the lowest number of ratings as the set of “new” users.

The numeric meta-parameters of the proposed models (i.e. thresholds, decay factors, etc.) were experimentally learnt by using a single validation set. We built the validation set by applying the per-user splitting protocol mentioned above on the original training set (using also 5 test ratings per user). To learn the optimal configurations efficiently we used the popular simplex search algorithm known as Nelder and Mead method [1965], which is used in many fields for meta-parameter optimization. Basically, this method begins with a set of points each of which represents a specific meta-parameter configuration and together form the initial simplex. At each iteration, the method performs a sequence of transformations of the simplex with the goal of finding a new configuration that decreases the prediction error on the validation set with respect to the previously evaluated configurations. In our experiments the algorithm converged, on average, after 15 iterations.

4.4.1. *Comparing User-Profile Learning Methods*

In this section, we evaluate the effectiveness of the proposed user-profile interest weighting methods described in Section 4.3.1. All these methods have in common that they learn the interest weights of a user based on the user’s ratings and the attribute-based item profiles. Therefore, their accuracy also depends on the precision of the item’s profile representation. In this experiment we weighted differently item’s attributes depending on its type: *tags* were weighted using the well-known TF-IDF scheme; *genres* based on the IDF weighted scheme (using the idea that less popular genres should be more informative); for the *directors* we employ a global rating-based popularity, and for the *actors* we used a measure of relevance based on actor popularity and their ranking positions in the movies that has acted. Finally, the weights of item profiles were normalized to the range [0, 1].

In this experiment we used as baseline user-profile weighting method the *rating average* proposed by Sen et al. [2009] and defined in Eq. 2.16. **Figure 4.4** illustrates a comparison of the accuracy improvement of the proposed methods (*CB-AdjRA* and *CB-Hybrid)* with respect to the baseline. In order to fairly compare the interest weighting methods, we use them in combination with the same CB prediction model based on dot product and defined in Eq. 4.4. *CB-AdjRA* is the variant using the adjusted *rating average* method defined in Eq. 4.1, and *CB-Hybrid*, the variant using the extended *rating average* method defined in Eq. 4.3.

It can be observed that both *CB-AdjRA* and *CB-Hybrid* outperform the baseline in both recommendation tasks: rating prediction (measured by RMSE and MAE) and ranking recommendation (measured by Recall and NDCG). Comparing the proposed variants, we can see that *CB-Hybrid* is clearly better than *CB-AdjRA*, especially for Recall and NDCG, where the improvement achieved is of 74% and 125%, respectively, in the “All” users set . This demonstrates that the extended variant contributes to estimate more precise user profiles and hence better item-to-user profile matching estimations.

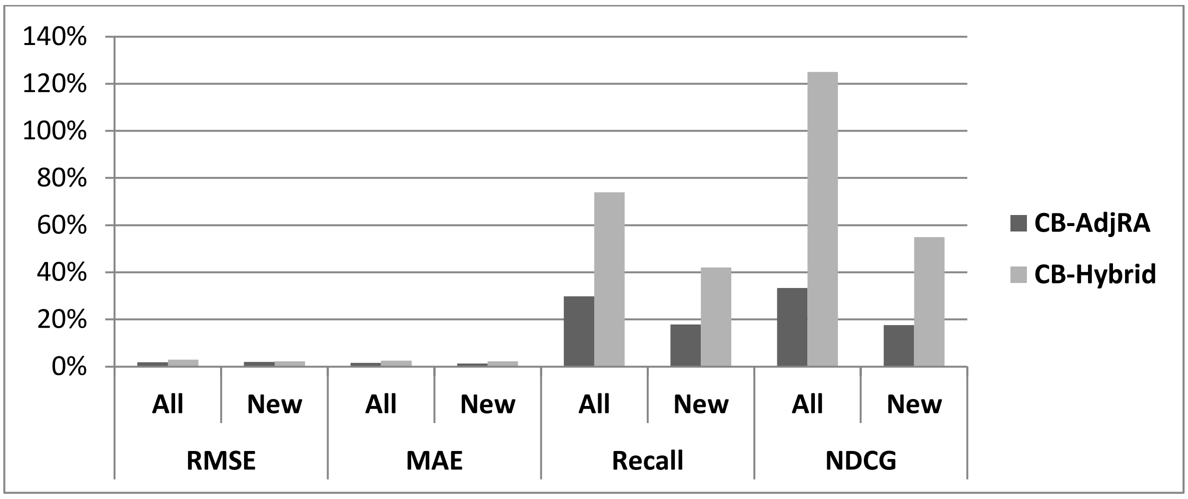


Figure 4.4. Accuracy improvement of the proposed user-profile learning methods (CB-AdjRA and CB-Hybrid) with respect to the baseline method. (*All* means that the result is averaged over all the users, and *New* averaged over the set of new users)

4.4.2. *Best-Pairs vs. All-Pairs Pairwise Strategies*

Here we compare the performance of the proposed SCB variants using several user-based distributional measures. In addition to the cosine latent-based measures described in Section 4.3.2, i.e. LSA and Forbes-Zhu method, we have included in the comparison a set-theory and a probabilistic measure. **Table 4.2** shows the results of the variant using the *best-pairs* strategy (*SCB-BP*) and the variant using the *all-pairs* strategy (*SCB-AP*). Both variants employ the best-performing user-profile learning method according to the results shown in previous section, i.e., the extended rating average method defined in Eq. 4.3. In this experiment, we use *CB-Hybrid* as the baseline algorithm. Regarding the distributional measures, *Jaccard* refers to the Jaccard index defined in Eq. 3.7, and *Kullback-Leibler* refers to the Kullback and Leibler [1951] probabilistic measure (defined in Eq. 3.9).

It can be observed that some of the distributional measures perform differently depending on the recommendation task (rating prediction or ranking) and the SCB variant. For example, the dimensionality reduction measures perform better than the others in terms of ranking precision when using *SCP-AP*: in general, *LSA* is the best measure, although *Forbes-Zhu* obtains slightly better results for new users. This implies that, if our idea is to use *SCP-AP* as part of a hybrid CF recommender, perhaps it is a better option to employ the *Forbes-Zhu* measure and thus focus more on providing better recommendations to new users, whereas if we use *SCP-AP* as the main recommendation technique, then *LSA* is the best option. Surprisingly, dimensionality reduction techniques do not perform as well for rating prediction. In this case, the probabilistic measure *Kullback-Leibler* is the one that achieves the best accuracy when used with *SCB-BP*. We experimentally found the latent dimensionalities that yielded better results, using 200 factors in *LSA*, and 90 factors in *Forbes-Zhu*.

Table 4.2. Prediction accuracy of SCB-BP and SCB-AP using user-based distributional measures. Results in bold are statistically significant better (95% confidence level) than the baseline. Results underlined are also significantly better than the non-underlined.

Models	Distributional Measure	RMSE		MAE		Recall		NDCG	
		All	New	All	New	All	New	All	New
CB-Hybrid	-	.85	1.00	.62	.85	.13	.18	.05	.08
SCB-AP	*Jaccard*	.85	1.00	.62	.85	**.17**	.15	**.07**	.06
	Kullback-Leibler	.95	1.16	.69	.89	**.16**	**.25**	.06	**.10**
	LSA	.85	1.00	.62	.85	<u>**.19**</u>	**.27**	**.07**	<u>**.12**</u>
	Forbes-Zhu	.85	.99	.61	.85	**.16**	<u>**.29**</u>	.06	<u>**.12**</u>
SCB-BP	*Jaccard*	.85	**.97**	.62	**.82**	**.17**	.16	**.07**	.06
	Kullback-Leibler	.84	<u>**.94**</u>	.61	<u>**.79**</u>	**.16**	.11	.06	.03
	LSA	.85	**.98**	.62	**.83**	**.18**	.18	**.07**	.07
	Forbes-Zhu	.84	**.98**	.62	**.83**	**.17**	.15	**.07**	.05

We compared the performance of both SCB variants when using the optimal distributional measures for each **recommendation task**, i.e., for **rating prediction** (*SCB-BP* + *Kullback-Leibler* and *SBP-AP* + *Forbes-Zhu*) and for **ranking** (*SCB-BP* + *LSA* and *SCB-AP* + *LSA*). **Figure 4.5** illustrates the % improvement of both SCB variants with respect to the CB baseline (*CB-Hybrid*).

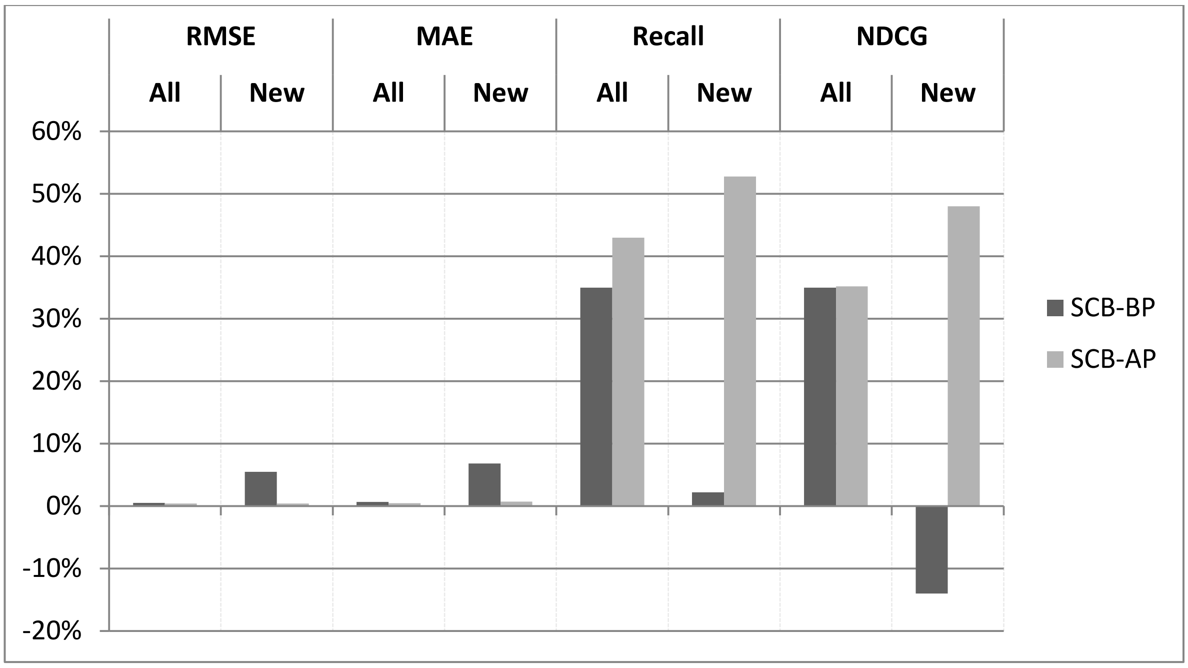


Figure 4.5. Accuracy improvement with respect to *CB-Hybrid* of *SCB-BP* and *SCB-AP* using the best-performing user-based distributional measure for each recommendation task

It can be observed that, for rating prediction, the improvements of both SCB variants are non-significant in the “All users" case, being only appreciable the improvement of *SCB-BP* for new users (8% gain). This demonstrates that the *best-pairs* strategy is better than the *all-pairs* one for rating prediction, and it is especially effective for users with small user profiles.

For ranking, the improvements of both variants are more significant. In this task, we can see that *SCB-AP* clearly outperforms *SCB-BP*, improving Recall and NDCG for both sets of users. The differences between variants are really significant for new users, which is when *SCB-BP* performs poorly compared to *SCB-AP*. In this case, *SCB-AP* achieves an improvement of 50% for Recall and NDCG with respect to *CB-Hybrid*. These results prove the usefulness of the *all-pairs* strategy for ranking, given that for this task what matters most is the order of the items and not the closeness between the predicted and the true rating.

We obtained the best results in *SCB-AP* when using a similarity threshold of 0.25, whereas in *SCB-BP*, the similarity threshold was set to 0 (the threshold is used to determine if a pairwise attribute combination is similar enough to be considered during score prediction). This low value of the thresholds in both variants means that almost all the positive semantic similarities derived by the distributional measures are useful for the prediction.

Regarding the selection of the aggregation function, our results show that the *product* is clearly the most effective aggregation function for rating prediction. But, for ranking prediction, it is not clear which aggregation method is better because it depends on the distributional measure and the pairwise strategy used. For example, the optimal SCB variant for ranking (i.e. *SCB-AP + LSA*) uses also the *product*, and in the case of *SCB-BP* with *LSA*, the best results are obtained when using the *average*.

4.4.3. *User-Based vs. Item-Based Distributional Semantics*

In this section we evaluate the effectiveness of *user-based* **distributional similarities**, which are based on co-occurrences of attributes over the corpus of user profiles, compared to the commonly used *item-based* **distributional similarities** (based on the co-occurrences of attributes with respect to items). Before comparing user-based and item-based similarities, we analyzed which distributional measure performs better with item-based ones: **Table 4.3** shows the results of *SCB-BP* and *SCB-AP* using the distributional measures that can be employed for item co-occurrence analysis, i.e. *Jaccard*, *Kullback-Leibler* and *LSA.* Differently from the results obtained when using *user-based* semantics, *Jaccard* is the measure that performs better for rating prediction when used with *SCB-BP*. For ranking with *SCP-AP*, *Jaccard* is also the best in “All” users, but *Kullback-Leibler* and *LSA* have better Recall for “New” users.

Table 4.3. Prediction accuracy of SCB-BP and SCB-AP using item-based distributional measures

Model	Distributional Measure	RMSE		MAE		Recall		NDCG	
		All	New	All	New	All	New	All	New
CB-Hybrid	-	.85	1.00	.62	.85	.13	.18	.05	.08
SCB-AP	*Jaccard*	.85	1.00	.62	.85	**.19**	.18	**.07**	.08
	Kullback-Leibler	.85	**.98**	.62	.84	**.16**	**.21**	.06	.09
	LSA	.85	.99	.62	.84	**.18**	**.21**	**.07**	.08
SCB-BP	*Jaccard*	.85	**.96**	.62	**.80**	**.17**	.16	**.07**	.06
	Kullback-Leibler	.85	1.00	.62	.85	**.17**	.13	**.07**	.06
	LSA	.85	**.98**	.62	.83	**.17**	.16	**.07**	.06

Figure 4.6 shows a comparison of the improvement, with respect to the baseline, of the best-performing SCB variants when using User-Based (UB) and Item-Based (IB) distributional similarities. When using item-based co-occurrences, the similarity between two attributes mainly depends on how-many items are annotated by both attributes. In contrast, when using the user-based perspective, the similarity depends on how-many users are interested in both attributes and the correlation of their interest weights. It can be observed that UB outperforms IB for rating prediction and ranking, and their differences are especially significant when dealing with new users. This demonstrates that the exploitation of user-based distributional semantics is more useful for overcoming the data-sparsity problem and improving CB recommendation.

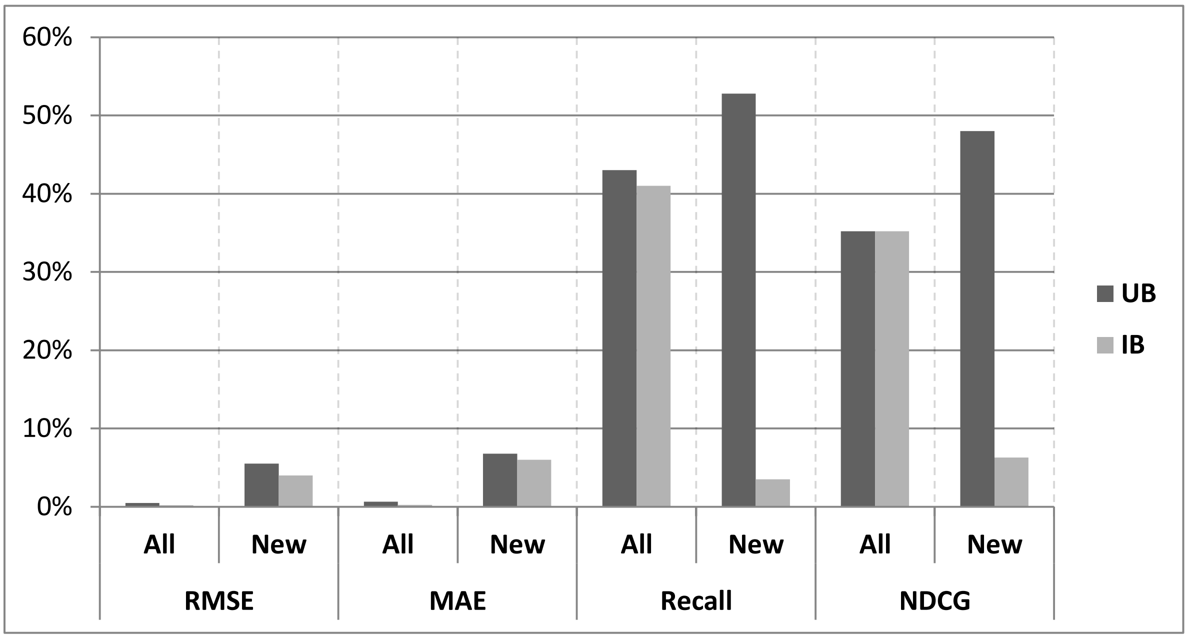


Figure 4.6. Accuracy improvement with respect to *CB-Hybrid* of the best-performing SCB variants for each task using User-Based (UB) and Item-Based (IB) distributional similarities

Table 4.4 shows an example of the type of similarities derived using each method in the MovieLens data set: a comparison of the top-5 attributes considered similar to the *marvel* user tag. As it can be observed, the top-5 similarities are completely different. In the item-based ranking the top-5 list is composed basically of actors and directors that specifically appear in movies that contain the tag *marvel*, but they are not necessary the most popular among users. In contrast, the user-based ranking is composed of three more general tags (*superhero, comic book adaptation*, and *legislation*), which in the specific domain of movies based on comics are strongly related attributes, such as the popular actor *Huge Jackman* and the director *Sam Raimi*. This demonstrates that similarities based on user co-occurrences "prioritize" semantically-related attributes that are popular among users.

Table 4.4. Top-5 similar attributes to attribute *marvel* (a user tag in the MovieLens data set) estimated using the user-based and item-based semantic vectors in combination with *LSA*. The value is the similarity value of the association (e.g. <*superhero*, *marvel*> pair similarity is equal to 0.85).

Item-based		User-based	
Doug Hutchison	0.82	superhero	0.85
Lexi Alexander	0.82	comic book adaption	0.83
Dominic West	0.74	legislation	0.82
Colin Salmon	0.72	Sam Raimi	0.82
John Stevenson	0.69	Hugh Jackman	0.81

4.4.4. *Distributional vs. Ontology-Based Measures*

In this section, we evaluate the effectiveness of the best-performing user-based distributional measure for each recommendation task (according to the performance results presented in previous section), by comparing it to the performance of well-known ontology-based measures. But, to perform this experiment we first needed a domain ontology defining the explicit relations between attributes of the MovieLens data set. In particular, we developed a movie taxonomy based on the movie genre classification publicly available on Amazon's website.

Figure 4.7 shows a partial representation of the Amazon's genre classification. We parsed the genre hierarchical relations using the Amazon Product Advertising API[13], which provides programmatic access to Amazon's product selection and discovery functionality in order to help users advertise Amazon products from their website. Using this API, we developed two programs: one responsible for crawling the Amazon's movie genre classification and creating a RDF file containing all the genres and their hierarchical relations; and the other responsible for automatically annotate the movies in the MovieLens

[13] See [https://affiliate-program.amazon.com/gp/advertising/api/detail/main.html], accessed on November 14th, 2013.

data set with the Amazon's genres. The ontology population was carried out by searching for movies in the Amazon catalog with the same or similar title to the one available on the MovieLens item content data. Using the developed crawler, we obtained a taxonomy of 420 genres and two levels of depth, and annotated 98% of the movies in the data set. In all these movies the Amazon's genres were included as additional attributes. Note that in the pruned MovieLens data set used for evaluation, we had already discarded the 2% movies not indexed in the Amazon's taxonomy.

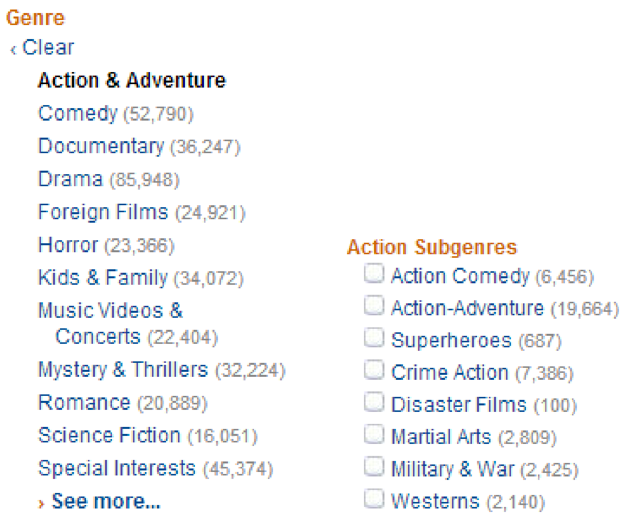


Figure 4.7 Partial view of Amazon's movie genre classification

As explained in Section 3.2.1, several ontology-based measures can be used to estimate the similarity between two hierarchically-related attributes. In particular, we included in the experimentation the following five ontology-based measures: *Wu-Palmer* (defined in Eq. 3.1), *Leacock-Chodorow* (Eq. 3.2), *Resnik* (Eq. 3.4), *Jiang-Conrath* (Eq. 3.5) and the *Lin*'s measure (Eq. 3.6).

Table 4.5 shows the results of *SCB-BP* and *SCB-AP* using the above mentioned ontology-based measures. The first thing one can observe is that no significant differences exist among variants. This can be explained by the limited richness of the taxonomy, which only has two levels of depth. This causes that the estimated similarity values almost do not differ among ontology-based measures. However, as in previous sections, results show that *SCB-BP* is slightly better than *SCB-AP* for rating prediction and new users, and *SCP-AP* outperforms *SCB-BP* in ranking task. Regarding the meta-parameters of *SCP-AP* and *SCB-BP*, we found the best performance when setting the *similarity threshold* to 0 in both cases, and used the *product* aggregation for rating prediction and the *maximum* for ranking.

Table 4.5. Prediction accuracy of SCB-BP and SCB-AP using ontology-based measures

	Similarity Measure	**Type**	**RMSE**		**MAE**		**Recall**		**NDCG**	
			All	**New**	**All**	**New**	**All**	**New**	**All**	**New**
CB-Hybrid	-	-	.85	1.00	.62	.85	.13	.18	.05	.08
SCB-AP	Wu-Palmer	Link-based	.84	.99	.61	.85	**.16**	**.29**	.06	**.12**
	Leacock-Chodorow		.84	.99	.61	.84	**.16**	**.28**	.06	**.12**
	Resnik	Node-based	.85	.99	.62	.84	**.16**	**.28**	.06	**.12**
	Lin		.84	.99	.61	.85	**.16**	**.29**	**.07**	**.12**
	Jiang-Conrath		.85	.99	.62	.84	**.16**	**.28**	.06	**.11**
SCB-BP	Wu-Palmer	Link-based	.85	**.98**	.62	**.83**	.12	**.26**	.05	**.11**
	Leacock-Chodorow		.85	**.98**	.62	**.83**	.12	**.26**	.05	**.11**
	Resnik	Node-based	.85	**.98**	.62	**.83**	.12	**.26**	.05	**.11**
	Lin		.85	**.98**	.62	**.83**	.12	**.27**	.05	**.12**
	Jiang-Conrath		.85	**.98**	.62	**.83**	.12	**.26**	.05	**.11**

Figure 4.8 illustrates the differences on performance of the best-performing SCB variants for rating prediction and ranking recommendation when using the user-based distributional similarities and the ontology-based similarities. In terms of rating prediction accuracy, we can observe that the *user-based* (UB) variant (which uses *SCB-BP* with *Kullback-Leibler*) outperforms the *ontology-based* (OB) variant (using SCB-BP with *Resnik*), reducing thus the prediction error by 4% for new users. In terms of ranking performance, the UB variant (using *SCB-AP* with *LSA*) outperforms the OB variant (using *SCB-AP* with *Lin*) in global terms ("All users"), improving its recall by 18%, and NDCG by 12%. In contrast, when recommending for new users the OB variant is slightly better, improving 64% recall and 53% NDCG with respect to *CB-Hybrid*, compared to the 53% and 48% gain achieved by the UB variant. However, as shown in previous section, when using *SCB-AP* with *Forbes-Zhu* (another user-based distributional measure evaluated) the obtained Recall and NDCG is almost the same as in the OB variant.

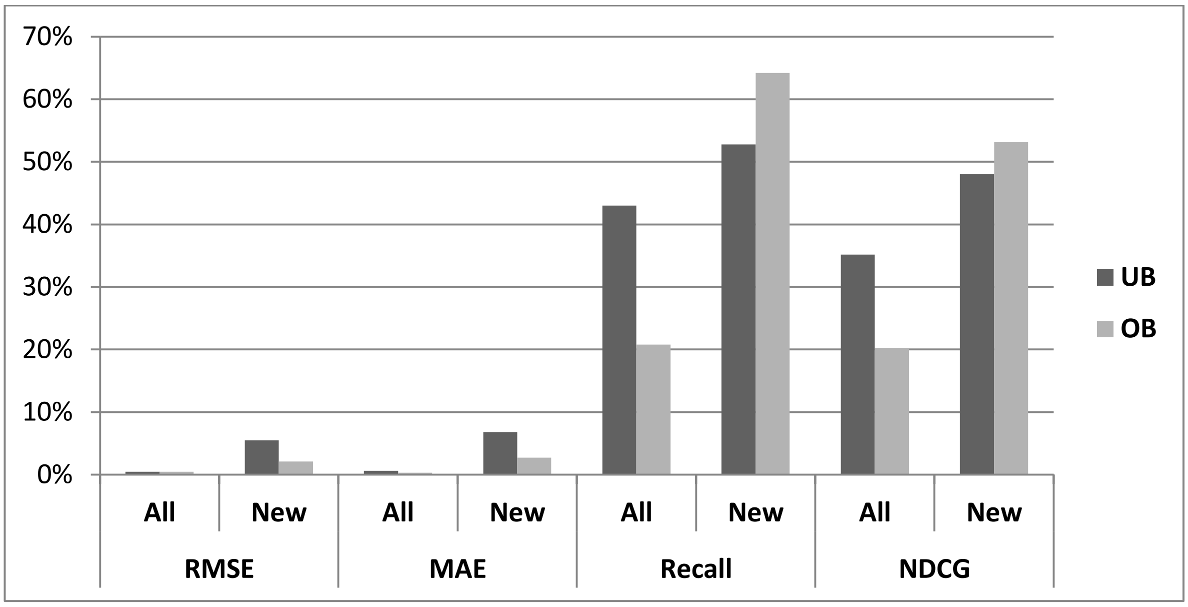


Figure 4.8 Accuracy improvement with respect to *CB-Hybrid* of the best-performing SCB variants using the user-based (UB) distributional measure and the ontology-based (OB) measure

4.4.5. *Pairwise Profile Matching vs. Profile Expansion*

Here we evaluate the performance of the proposed pairwise profile matching strategies compared to a CSA strategy for user profile expansion. In particular, we adapted, as explained below, a version of the CSA strategy proposed by Cantador et al. [2008; 2011] that was especially designed for working on ontologies with hierarchical and domain-specific relations.

In the original CSA strategy the authors use three constraints to limit the propagation of the interest weights: a threshold that delimits the maximum number of propagation steps in the graph (the *expansion* threshold), a threshold delimiting the minimum interest weight to be propagated (the *activation* threshold) to the associated concepts, and a threshold delimiting the fan-out (i.e. the “hub” effect). Given the different nature of distributional similarities in which the estimated similarity value is especially relevant to determine whether the weight must be propagated or not (i.e. some positive semantic associations can be too weak to be used for interest propagation), we have included an additional threshold that delimits the minimum attribute-to-attribute similarity to be considered as a relevant association (the *similarity* threshold) and hence to be used for propagation. In this case, we did not use the fan-out threshold.

Figure 4.9 shows the pseudo-code of the implemented CSA algorithm. The input of the algorithm is an initially learnt user profile and a specific configuration of above mentioned thresholds. The output consists of a modified user profile as the result of applying the semantic propagation of interest weights. Initially, all the concepts of the user profile with interest weight larger than the *activation* threshold are added in a queue sorted by activation value. Then, starting by the concept with larger activation value of the queue, we propagate its value if the current number of propagation steps is smaller than the *expansion* threshold. In this case, the propagation is applied to all the strongly associated attributes (whose list is returned by the *getSimilarConcepts*(a, t_s) function and consist of the attributes whose similarity value is larger than the *similarity* threshold). The spreading process is executed until the queue is empty.

```
Input
  $w_u$: initial user profile
  $t_e$: expansion threshold
  $t_a$: activation threshold
  $t_s$: similarity threshold
Output
  $w'_u$: semantically-expanded user profile

  queue = getInitialPriorityQueue($w_u$,$t_a$);
  while queue.Count > 0 do
    $a$ = queue.Poll();
    level = getPropagationLevel($a$);
    if (level < $t_e$) then
      foreach $a^* \in$ getSimilarConcepts($a$, $t_s$) do
        $w'_{ua^*} \mathrel{+}= (1 - w'_{ua^*}) \times w_{ua} \times sim(a, a^*)$;
        setPropagationLevel($a^*$,level+1);
        queue.Add($a^*$);
      endfor
    endif
  endwhile
return $w'_u$
```

Figure 4.9. Implemented *Constrained Spreading Activation* strategy

Note that the algorithm only propagates the weight of positive interests, and the sum of propagated values can be at most 1 because of the following propagation formula:

$$w'_{ua^*} \mathrel{+}= (1 - w'_{ua^*}) \times value \tag{4.7}$$

where w'_{ul} is the interest weight of user u in attribute a^* before the propagation, and *value* refers to the propagated weight from the source attribute a, and it is calculated as w_{ua} multiplied by $sim(a, a^*)$;

Before comparing the effectiveness of the CSA strategy to the proposed pairwise strategies, we first analyzed the performance of CSA with respect to the distributional measures. Note that the CSA strategy is used to semantically expand the user profiles initially learnt by using the best-performing user-profile learning method (defined in Eq. 4.3). Then, recommendations are produced by means of the item-to-user profile matching based on dot product defined in Eq. 4.4.

Table 4.6 shows the performance results of *SCB-CSA* (the variant of SCB using the developed CSA strategy and whose algorithm is defined in **Figure 4.9**) in combination with several user and item-based distributional measures. Similarly to the pairwise strategies (*SCB-BP* and *SCB-AP*), it can be observed that there is no clear dominant measure, and the best-performing one again depends on the recommendation task (rating prediction or ranking). For rating prediction on new users, *SCB-CSA* with

Kullback-Leibler performs better than the rest, but in this case, when using item-based (IB) co-occurrence analysis rather than user-based one, as in *SCB-BP*. And for ranking, the best measure is user-based (UB) with *LSA*, as in *SCB-AP*. Regarding the meta-parameters of *SCB-CSA*, we experimentally obtained the best-performing configuration, which was the following: $t_e = 1$; $t_a = 0.4$; $t_s = 0.25$.

Table 4.6. Prediction accuracy of *SCB-CSA* using user- and item-based distributional measures

	Similarity Measure	Perspective	RMSE		MAE		Recall		NDCG	
			All	New	All	New	All	New	All	New
CB-Hybrid	-	-	.85	1.00	.62	.85	.13	.18	.05	.08
SCB-CSA	Jaccard	IB	.84	.99	.62	**.83**	.14	.17	.05	.08
		UB	.90	1.12	.64	.94	.12	.13	.04	.06
	Kullback-Leibler	IB	.86	**<u>.97</u>**	**.61**	**.82**	.13	.17	.05	.07
		UB	.90	1.11	.64	.94	.12	.13	.04	.05
	LSA	IB	.85	.99	.62	**.83**	.14	.16	.05	.06
		UB	.90	1.12	.64	.94	**<u>.19</u>**	.19	**<u>.07</u>**	.09
	Forbes-Zhu	UB	.86	.99	.62	**.82**	.12	.14	.04	.04

Figure 4.10 illustrates the differences on performance of three SCB variants: *SCB-BP* (using the *best-pairs* strategy with UB semantics and *Kullback-Leibler*), *SCB-AP* (using the *all-pairs* strategy with UB semantics and *LSA*), and *SCB-CSA* (using the adapted CSA strategy with IB and *Kullback-Leibler* for RMSE and MAE, and with UB and *LSA* for Recall and NDCG).

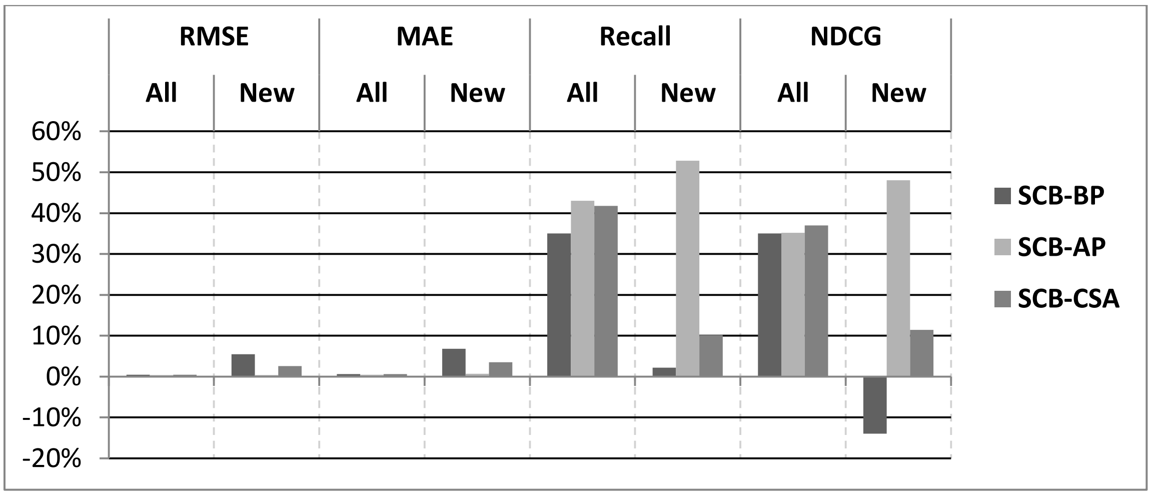


Figure 4.10. Accuracy improvement with respect to CB-Hybrid of three SCB variants: using the best-pairs (BP) strategy, using the all-pairs (AP) strategy and using the CSA strategy.

The bar charts show that for new users ("New columns") *SCB-BP* outperforms *SCB-CSA* in terms of RMSE and MAE, and *SCB-AP* outperforms *SCB-CSA* for Recall and NDCG. According to these results we can conclude that *SCB-CSA* performs worse than the SCB variants using the proposed pairwise strategies. We also evaluated a SCB variant combining the CSA strategy and pairwise strategies, but similarly to the experimentation presented in Section 3.4, this type of combinations did not obtain significant improvements, and in some cases even worse results.

4.4.6. *Performance Compared to the State of the Art*

In this section, we evaluate the performance of *SCB-BP* and *SCB-AP* using the best-performing user-based distributional measures with respect to the following state-of-the-art SERS and CF techniques:

- *CB-LSA* [Bambini et al. 2011], a SERS technique that uses LSA to directly reduce the original items' representation, and then learns the interest weights based on the same latent semantic space using a folding-in method (see section 3.3.1 for more details about this approach). The rating estimation function is based on the dot product of the latent factor representations of item and user profiles in addition to the user and item biases (as defined in Eq. 2.17).
- *SVD++* [Koren and Bell 2011], which is a well-known MF model optimized for rating prediction using SGD. Its rating estimation function is defined in Eq. 2.26.
- *BPR-MF* [Rendle et al. 2009], which is a MF model optimized for ranking based on the BPR framework (see Eq. 2.27 in section 2.4.3 for an exact definition of this model).

Figure 4.11 illustrates a performance comparison of the models considered based on the % improvement with respect to *CB-Hybrid*.

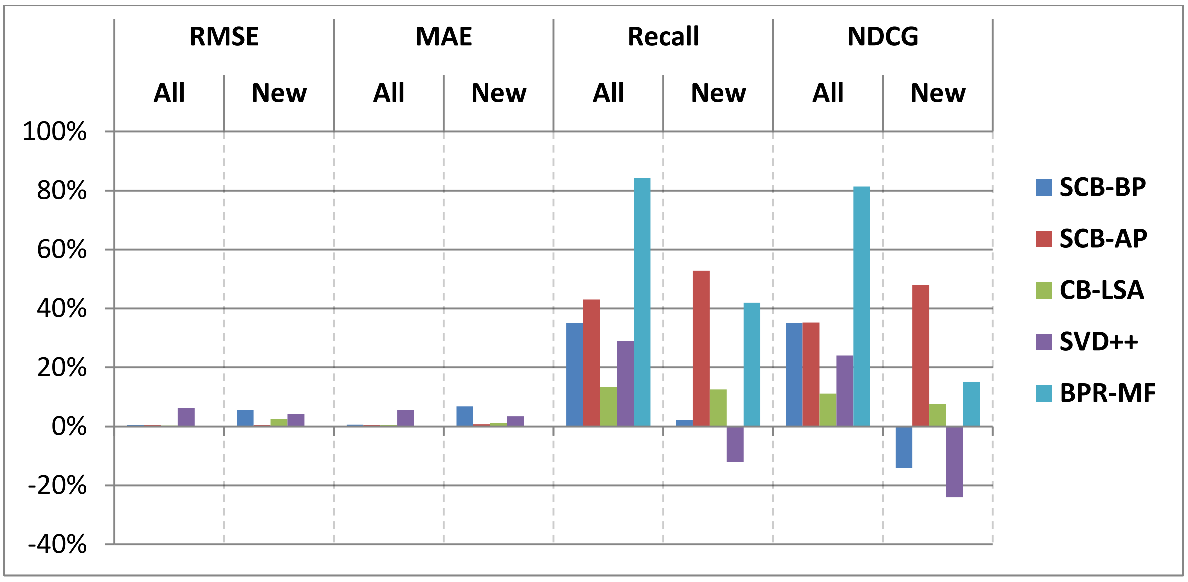


Figure 4.11. Accuracy improvement with respect to *CB-Hybrid* of the proposed SCB variants and the state-of-the-art SERS and CF techniques

First we analyze the results of the SCB variants with respect to *CB-LSA* (the SERS technique using an item-based distributional measure). It can be observed that *SCB-BP* outperforms *CB-LSA* in RMSE and MAE, improving its results by 4% on new users (“New” column). In terms of Recall and NDCG, CB-LSA is also clearly outperformed by *SCB-AP* in both sets of users (“All” and “New” columns).

Compared to the MF models, *SCB-BP* outperforms SVD++ when predicting ratings to new users, improving its results by 4%, and *SCB-AP* is also better than SVD++ for ranking in both sets of users. Similarly to other researchers that investigated recommendation models directly optimized for ranking [Rendle et al. 2009; Weimer et al. 2009], our results also confirm that models optimized for RMSE or MAE, like *SVD++*, do not necessarily perform well for ranking. Note that SVD++ even has worse Recall and NDCG than *CB-Hybrid* on new users. Finally, *SCB-AP* also outperforms *BPR-MF* in the new users set, improving its recall by 10% and NDCG by 30% with respect to the baseline. However, both state-of-the-art MF models are clearly superior to SCB variants in the “All” users set: *SVD++* improves the baseline by 6% compared to the 1% achieved by *SCB-BP*, and *BPR-MF* improves *CB-Hybrid* by 80% compared to the 40% achieved by *SCB-AP*.

Due to the small improvements achieved by SCB variants in terms of RMSE and MAE in the “All” users set, we extended our experimentation by including another state-of-the-art method for user-profile learning, which learns weighted user profiles optimized for RMSE using SGD. The experiment consisted of extending with semantics this optimized CB model incorporating the proposed pairwise strategies, and evaluating the improvement achieved. Particularly**,** we have included in this performance comparison the following two SGD-based CB prediction models:

- *CB-SGD*, which learns the optimal user profiles by using SGD without exploiting distributional semantics. As explained in section 2.3.4, using the SGD method, the interest profiles w_u are incrementally learnt together with the user and item biases. Using the rating estimation function defined in Eq. 4.4, this model optimizes the model parameters minimizing the following objective function:

$$\min_{b_*w_*}\sum_{r\in R}\left[\left(r_{ui}-\mu+b_u+\ b_i+\frac{1}{|M_{ui}|^{\alpha}}\,w_i^T w_u\right)^2+\lambda\left(b_u^2+\ b_i^2+\|w_u\|^2\right)\right] \quad (4.8)$$

- *SCB-SGD-BP*, which extends *CB-SGD* by incorporating the *best-pairs* strategy to the estimation function in both the learning phase using SGD and the prediction phase. Therefore, it learns the model parameters optimizing the following objective function (BestPairs(w_i, w_u) refers to the best-pairs profile matching defined in Eq. 4.6):

$$\min_{b_*w_*}\sum_{r\in R}\left[\left(r_{ui}-\mu+b_u+\ b_i+\frac{1}{|M_{ui}|^{\alpha}}\mathrm{BestPairs}(w_i,w_u)\right)^2+\lambda\left(b_u^2+\ b_i^2+\|w_u\|^2\right)\right] \quad (4.9)$$

We did not include a variant using the *all-pairs* strategy because we have previously demonstrated that, for rating prediction, it does not perform well and in this experiment we aimed at improving the rating prediction accuracy of *CB-SGD*. **Figure 4.12** illustrates the RMSE reduction with respect to *CB-Hybrid* of the considered models, including also *SCB-BP* and *SVD++*.

As expected, the CB variant optimized for RMSE (*CB-SGD*) clearly outperforms *CB-Hybrid* and *SCB-BP* for RMSE in the “All” users set, reducing the error by 2% with respect to *CB-Hybrid*. However, we can see that in the “New” users set, *SCB-BP* (which employs the same user-profile learning method as *CB-Hybrid*) outperforms *CB-SGD*. This demonstrates that the SGD-based user profile learning methods is especially effective when enough rating data are available but not as good as *SCB-BP* when dealing with cold-start (new) users.

Analyzing the improvement achieved by the SGD-based SCB variant (SCB-SGD-BP), it can be observed that it slightly outperforms *CB-SGD* in the “All” users set, reducing RMSE by almost 1%, but not when making predictions to new users. This was an unexpected result given that *SCB-BP* is especially effective on new users. A possible explanation to this poor performance of SCB-SGD-BP for new users is that, during model learning, the *best-pairs* strategy increases the overfitting effect in those users with few training ratings.

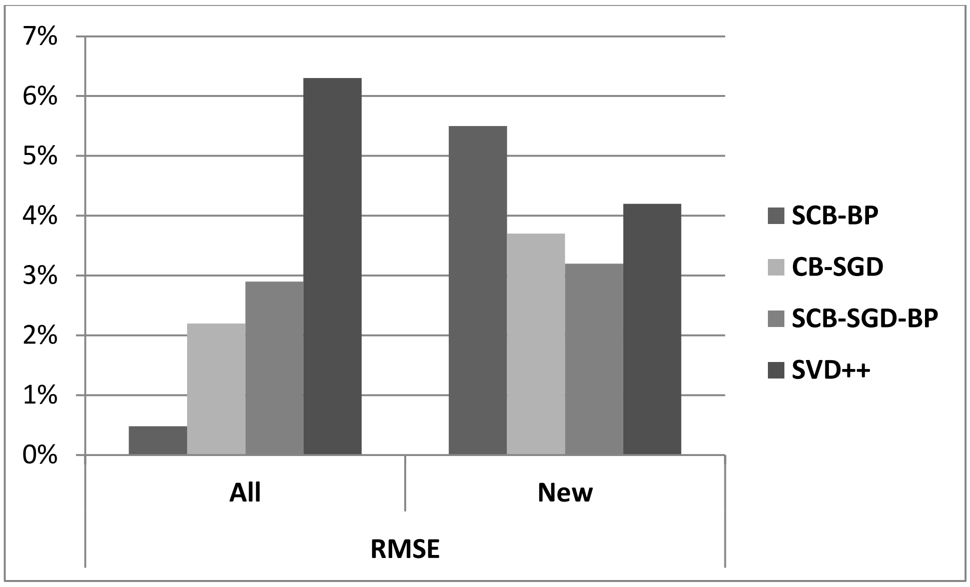


Figure 4.12. RMSE reduction of SGD-based CB and SCB variants with respect to *CB-Hybrid*

4.5. Conclusions

In this chapter we have proposed *Semantic Content-Based* (SCB) filtering, a novel SERS approach to CB recommendation that exploits distributional similarities between item's attributes by incorporating pairwise profile matching strategies in the prediction phase. In particular, we described two SCB variants: one using a *best-pairs* strategy that is more suitable for rating prediction, and another one using an *all-pairs* strategy, more suitable for ranking recommendation, where the exact predicted scores are not as relevant as the order of the recommended items.

We have presented an exhaustive evaluation of the proposed SCB variants using a well-known movie rating data set including also item content data. We have analyzed their performance using several distributional measures, demonstrating than similarities between items' attributes derived from user-based co-occurrences are more effective for improving prediction accuracy than item-based distributional similarities or ontology-based similarities. We have also demonstrated that the proposed variants based on pairwise matching strategies clearly outperform the commonly used profile expansion technique using spreading activation as well as to one strategy for latent item-profile representation using LSA. Finally, compared to the state-of-the-art MF models, we have shown that SCB variants are better when recommending to new users.

Chapter 5 - Exploiting Distributional Semantics for enhanced Context-Aware Recommendation

5.1. Introduction

The main goal of Context-Aware Recommender Systems (CARSs) is to enhance the effectiveness of CB and CF recommendation techniques by incorporating contextual information in the recommendation process. The success of CARSs depends basically on two factors: (1) how relevant the context captured by the system is in a particular domain; (2) how sparse the user training data are, since context-aware prediction models require a large number of contextually-tagged ratings provided in all the various contextual situations that may be encountered by a user while experiencing an item.

As shown in previous chapters, the exploitation of semantic similarities between item's attributes is a good solution to mitigate the sparsity-related limitations of CB recommendation techniques. Likewise, semantics about contextual information can be exploited during context modeling to mitigate the data-sparsity problem and thus enhance the performance of CARS techniques. In particular, this knowledge can be useful to reuse user data tagged with syntactically different contexts but semantically similar. For instance, if we want to predict the rating for a *museum* and the target contextual situation includes a condition such as "group composition is two adults and two children", ratings acquired when the group composition was "two adults and three children" might be used, too, to generate a more robust and accurate predictive model in the target situation.

In Chapter 4 we demonstrated that the exploitation of distributional semantic similarities derived from co-occurrences over user profiles is an effective method to improve the prediction accuracy of CB recommendation techniques, especially in cold-start situations. Similarly, here we want to demonstrate that distributional semantics of contextual conditions, derived from contextually-tagged user rating data, can help to improve the prediction accuracy of context-aware recommendation. Following with the same example in the tourism domain, if we analyze how the conditions "sunny" day and group of "two adults and two children" influence the users' ratings, we may discover that they actually have a similar influencing pattern: they both tend to increase the user's ratings for outdoor places like castles, and decrease them for indoor places like museums. Based on the similarity of their influence patterns we might consider them as semantically similar.

This chapter presents a reduction-based pre-filtering approach, called *Semantic Pre-filtering* (SPF), which exploits similarities between situations based on the distributional semantics of contextual conditions, i.e., assuming that two situations are similar if they are defined by elementary contextual conditions that influence users' ratings in a similar way. Given a target contextual situation, SPF uses ratings tagged with contextual situations similar to the target one to generate a more precise local prediction model for recommending in the target context. In order to determine if a candidate situation is similar enough to the target one, SPF uses a global *similarity threshold* that specifies the minimum semantic similarity required for situations to be reused: the larger the threshold, i.e., the higher the required similarity is, the sharper the contextualization is. This implies that fewer situations are used to build the rating prediction model adapted to the target contextual situation. For that reason, we say that the predictive model is (more) “local”.

This chapter is organized as follows. Section 5.2 positions our work with respect to the state of the art. Section 5.3 presents the details of SPF and the proposed variants. Section 5.4 presents an extensive offline evaluation of SPF using several contextually-tagged data sets as well as a detailed analysis of the effect of the similarity threshold on the system’s performance. Finally, Section 5.5 summarizes the work presented in this chapter and highlights the main conclusions.

5.2. Related Work

CARSs are generally classified into three paradigms [Adomavicius and Tuzhilin 2011; Adomavicius et al. 2011]: *contextual pre-filtering*, where context is used for selecting a set of contextually relevant rating data that are exploited for generating a target context-dependent recommendation (using a context-free model); *contextual post-filtering*, where context is used to adjust (filter) recommendations generated by a context-free model; and *contextual modeling*, in which contextual information is directly exploited in the adopted context-aware recommendation model.

Among these paradigms, *pre-filtering* is especially appealing because it has a straightforward justification: when context matters, use in the recommendation process only the rating data acquired in the same contextual situation of the target user, because only these ratings are relevant for predicting user preferences in that context. However, because of this ratings’ filtering process, a major limitation of pre-filtering approaches is the *data-sparsity* problem. Currently, two major pre-filtering approaches have been proposed in the literature: *reduction-based* [Adomavicius et al. 2005] and *splitting* approaches [Baltrunas and Ricci 2009; 2014; Zheng et al. 2013a] (See Section 2.5.2 for a description of pre-filtering approaches).

Approaches based on *contextual modeling* extend context-free predictive models by explicitly modeling the influence of context on the rating prediction, i.e., by typically adding new parameters that represent the contextual information. Currently, two major approaches based on extending Matrix Factorization (MF) prediction models can be identified in the literature: *Tensor Factorization* (TF) [Karatzoglou et al. 2010; Rendle et al. 2011] and *Context-Aware Matrix Factorization* (CAMF) [Baltrunas et al. 2011b; 2012; Odić et al. 2013; Braunhofer et al. 2014] (See Section 2.5.4 for a description of contextual modeling approaches)

Recent empirical results indicate that no universal context-aware approach exists, and the best performing method depends on the recommendation task and domain [Panniello et al. 2009; 2010; 2014]. Among the findings, their analysis shows that the accuracy of all considered CARS techniques decreases when the contextual information has a finer granularity and fewer ratings tagged with the target situation are available. In this thesis, we tackle the sparsity-related limitations of the state of the art by exploiting distributional similarities between contextual situations during context modeling.

In the literature, few SERS techniques to context-aware recommendation have been proposed, and the existing ones only exploit semantics of contextual conditions derived from context taxonomies [Adomavicius et al. 2005; Liu et al. 2010; Bouneffouf et al. 2012] (See section 3.5 for a description of these three approaches). As commented previously, ontology-based approaches are limited by the quality of the context ontology used, which may not suit the data.

Semantic Pre-filtering, the method proposed in this chapter, is analogous to *Generalized Pre-filtering*, proposed by Adomavicius et al. [2005], because it is also a reduction-based approach, but instead of searching for the optimal segmentation of the ratings, it exploits distributional similarities between situations to generate segments that aggregate the ratings tagged with situations similar to the target one. Therefore, the key difference is that our approach employs a notion of situation-to-situation similarity based on the distributional semantics of contextual conditions instead of relying on the usually limited condition-to-condition hierarchical relationships defined in the context taxonomy. As we will show later, this novel notion of similarity supports a more flexible and effective aggregation of the ratings and thus yields a better accuracy with respect to the state of the art, especially when the contextual situations considered in the application are very specific, i.e., defined by the conjunctions of several contextual conditions.

5.3. Semantic Pre-Filtering

To better understand SPF, it is worth recalling how the reduction-based pre-filtering method operates. When it is requested to compute recommendations in a target contextual situation, reduction-based pre-filtering follows a two-step process: firstly, a subset of training ratings relevant to that contextual situation are selected from the training set; secondly, a predictive model is built based on the collected ratings, which is then used to make predictions to users exactly in that situation. We say that this model is “local” because it is not based on the full set of available ratings but exploits only a subset of more relevant ratings.

The key step of this process is therefore the selection of the ratings that must be estimated to be relevant to the target contextual situation. In SPF, in addition to the ratings acquired exactly in the target situation, ratings acquired in situations “similar” enough to the target one are also used. SPF uses a custom definition of similarity that will be described in Section 5.3.2. Then, the selection of the “similar” contextual situations is determined by a **similarity threshold** (*t*), which is a global parameter that must be tuned to the data set; it determines the minimum similarity score between two situations to make one reusable when the target contextual situation is defined by the other. The larger the threshold is and the closer to 1 is (maximum similarity), the less contextual situations are selected, and consequently the more the rating predictive model is adapted to the target contextual situation. In particular, when t=1 the predictive model is equal to the one built by *Exact Pre-filtering* (a reduction-based variant proposed by Adomavicius et al. [2005]: only the ratings acquired in the target contextual situation are used).

As in other machine learning tasks, it may not be the case that a model fitting exactly the available training data (the ratings provided in the target contextual situation) provide the best predictions on future data, i.e., not used to train the model: overfitting the local contextual situations may jeopardize the overall system behavior, especially when ratings data are scarce. Therefore, one must detect the optimal middle point between a global model based on all user ratings (i.e. a context-free model) and a strict local model, which is just fitting the user ratings in a specific context. In SPF, we find the right level of contextualization for a given data by learning the similarity threshold that maximizes rating prediction accuracy.

Making more precise the above discussion, given a target contextual situation s^* and a similarity threshold *t*, in SPF the local training set R_{s^*}, which is the set of the ratings acquired in situation s^*, is expanded by adding all the ratings acquired in all the situations *s* where $Sim(s, s^*) \geq t$. This expanded training set is then used for building the local rating prediction model for the target situation s^*. Denoting with R_s the set of ratings acquired in situation *s*, the set X_{s^*} of the training data for the local model of s^* is:

$$X_{s^*} = \bigcup_{s:\, Sim(s,s^*) \geq t} R_s \tag{5.1}$$

Figure 5.1 illustrates the generation of the training data for a target contextual situation in SPF. In this example, it is shown that only the sets of ratings R_S tagged with a contextual situation whose similarity with the target one is larger than the similarity threshold (*t*) are selected as relevant for learning the local predictive model. In this example, only the ratings tagged with s_1 are selected. We note that, if more than one rating for a given user and item are available in some contextual situation similar to the target one, the average of these ratings is computed in order to generate a unique rating for a given user, item and contextual situation. Using this procedure we reduce the original multi-dimensional rating data to a two-dimensional rating matrix that can be then used for learning a local predictive model based using any of the available context-free recommendation techniques.

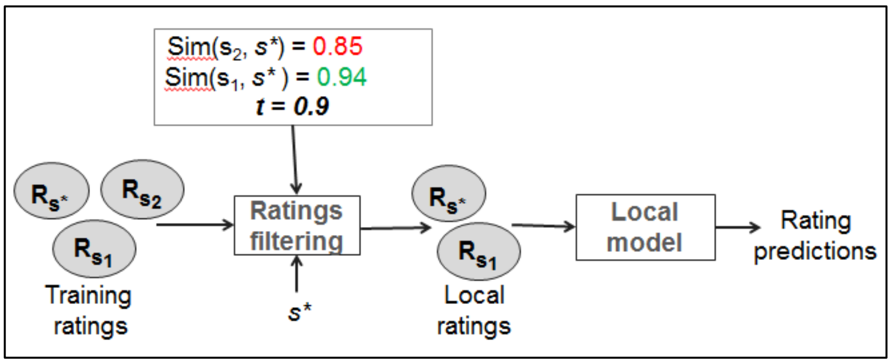


Figure 5.1. Example of the ratings selection process in SPF

5.3.1. *Distributional Semantics of Contextual Conditions*

In the previous section, we have assumed the existence of a similarity function between contextual situations that can be used to determine which ratings are actually used to generate context-aware recommendations for any given target contextual situation. This section describes how a viable distributional situation-to-situation similarity function can be computed based on the distributional semantics of contextual conditions with respect to the ratings, i.e., based on the *distributional hypothesis*, which is rooted in the assumption that two situations are similar if their composing conditions influence users' ratings in a similar way. To represent the distributional semantics of contextual conditions we use a semantic-vector space (VSM) with the goal to define the similarity between conditions in terms of their proximity in a high-dimensional vector space.

We propose to model a contextual condition by describing its influence on the average rating of the items or the users of the system. Therefore, the dimensionality of the resulting semantic vectors is equal to either the number of items or users. In more detail, our method exploits rating information to measure the influence of a condition as the aggregated deviation between the observed ratings when the condition holds (r_{uic}), and the predicted context-free rating ($\hat{r}_{ui}$). If we use a *user-based* perspective, then the influence of a condition *c* on a user *u*, which is denoted by w_{cu}, is calculated as follows:

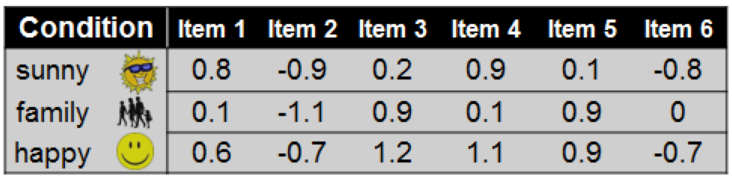
$$w_{cu} = \frac{1}{|R_{uc}| + \beta} \sum_{r_{uic} \in R_{uc}} (r_{uic} - \hat{r}_{ui}) \tag{5.2}$$

where R_{uc} is the set of ratings of the user *u* in condition *c*; and β is a decay factor used to reduce the value of the influence when $|R_{uc}|$ is relatively small based on the assumption that the larger the number of ratings, the more trustworthy is the above deviation estimate. In our experiments, we obtained the best results when using a value of $\beta \in [0,10]$; the exact value depends on the data set. As context-free predictive model we used the baseline predictor presented in Section 2.2 that optimizes the item and user biases by means of SGD and whose prediction formula is defined in Eq. 2.1. We also tested more sophisticated context-free predictive models but no significant performance differences were observed.

Likewise, the measure of the impact of a contextual condition can also be based on the effect of the condition on the ratings for an item. If R_{ic} denotes the set of ratings for item i in condition c, then in the *item-based* perspective the influence of the condition c on the item i, which is denoted by w_{ci}, is defined as follows:

$$w_{ci} = \frac{1}{|R_{ic}| + \beta} \sum_{r_{uic} \in R_{ic}} (r_{uic} - \hat{r}_{ui}) \tag{5.3}$$

Using the above mentioned formulas, we can build the semantic-vector representation of each condition with respect to the items or users in the training set. **Figure 5.2** illustrates an example with the semantic vectors of three contextual conditions with respect to six items. In such representation, a positive value means that the condition tends to increase the ratings given to the item, a negative value means that the condition tends to decrease those ratings, and zero indicates that the condition does not have any effect on the item ratings based on the rating data available: the larger the value, the larger the impact of the condition.



Condition	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
sunny	0.8	-0.9	0.2	0.9	0.1	-0.8
family	0.1	-1.1	0.9	0.1	0.9	0
happy	0.6	-0.7	1.2	1.1	0.9	-0.7

Figure 5.2. Semantic vectors of three conditions with respect to six items

When data are sparse, the generated semantic vectors could contain many values close to zero, which can lead to wrong similarity estimations. In this situation, analogously to other distributional measures in the literature (see Section 3.2.2), we propose to apply dimensionality reduction techniques to produce a more compact and informative semantic representation. In this work, we have experimented two techniques: (1) by applying the conventional SVD to the original co-occurrence matrix (as in LSA), and (2) by measuring the influence of a contextual condition on groups of items or users defined by a category attribute (i.e. types of items or users). We detail in the following these two techniques.

The reduction technique based on SVD decomposes the original $o \times m$ condition-to-item or condition-to-user co-occurrence matrix $A = [w_{ci}]$ (or $A = [w_{cu}]$) into three smaller matrices $A = U \times S \times V^T$ with U an $o \times r$ matrix, S an $r \times r$ diagonal matrix, and V an $m \times r$ matrix, such that U and V are orthogonal and r is the rank of the original matrix A. Then, the best rank-k approximation of A, which is denoted with A_k, is produced by selecting the k largest singular values in S and set the others to zero (i.e. $A_k = U_k \times S_k \times V_k^T$). Finally, we obtain the k-dimensional latent semantic vectors of each condition multiplying the rows of U_k by the square root of S_k. In our experimentation, we used the Lanczos algorithm of SVDLIB because it has been proved to be a precise and efficient method compared to other approaches [Kurucz et al. 2007]. (See Section 2.4.3 for more details).

The second reduction technique that we propose relies on an explicit grouping of the items according to the values of an attribute of the items or users. We call these values categories. For instance, one may know that the recommended items can be classified into movies of different genres (categories), or the users may be classified into age class categories. Using this grouping into different categories it is possible to estimate the influence of a contextual condition (on average) on all the items or users in the same group (same category). This makes sense if the contextual conditions have a uniform impact on all the items or users in the same group. In this work, we only experimented with item categories because this was the only information available in the data sets used in the evaluation. Denoting with R_{gc} the set of ratings tagged with the contextual condition *c* and associated to items in category *g*, then the influence of a condition *c* with respect to a category *g*, which is denoted by w_{cg}, is defined as follows:

$$w_{cg} = \frac{1}{|R_{gc}| + \beta} \sum_{r_{uic} \in R_{gc}} (r_{uic} - b_{ui}) \tag{5.4}$$

5.3.2. *Situation-to-Situation Similarity*

Relying on the previously described semantic representation of contextual conditions, we can measure the semantic similarity between two contextual conditions and between two generic contextual situations. We recall that a contextual situation is defined by the conjunction of one or more conditions (e.g. a contextual situation may be defined by temperature=*hot*, season=*summer* and mood=*happy*). In this chapter, we evaluate three strategies to measure situation-to-situation similarity based on the distributional semantics of conditions: (1) a *best-pairs* strategy, which aggregates the best-matching pairwise similarities of the conditions belonging to the compared situations, (2) an *all-pairs* strategy, which aggregates all the possible pairwise similarities of the conditions belonging to the compared situations; and (3) a *direct* strategy, which directly measures the similarity of two situations by representing the situations, similarly to the conditions, as vectors of influence scores on the items or users (i.e. semantic vectors), and then computes the *cosine* similarity.

If the compared situations are defined by only one condition, we define the situation-to-situation similarity as the condition-to-condition similarity between the candidate c and the target condition c^*, which we calculate as the cosine of the angle between their respective semantic vectors, denoted by w_c and w_{c^*} respectively (*l* is the dimensionality of the semantic representation of a contextual condition):

$$sim(c, c^*) = \frac{w_c^T w_{c^*}}{\sqrt{\sum_{i=1}^{l} w_{ci}^2} \times \sqrt{\sum_{i=1}^{l} w_{c^*i}^2}} \tag{5.5}$$

If one of the compared situations is defined by several conditions, in the *best-pairs* strategy we define the similarities between a target situation s^* and a candidate situation s by comparing, for each contextual condition c in the candidate situation, the most similar condition in the target situation (i.e. $\max_{c^* \in s^*}$). This is the *best-pairs* aggregation method:

$$sim(s, s^*) = \frac{1}{|s|} \sum_{c \in s} sim(c, \max_{c^* \in s^*}) \tag{5.6}$$

In the *all-pairs* strategy, we define the similarity between a target situation s^* and a candidate situation s by comparing all the condition pairs in the two situations. The *all-pairs* aggregation method is defined as follows:

$$sim(s, s^*) = \frac{1}{|s| \times |s^*|} \sum_{c \in s} \sum_{c^* \in s^*} sim(c, c^*) \tag{5.7}$$

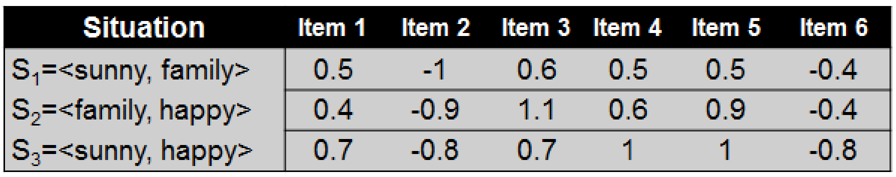
Differently from the previous strategies, in the *direct* strategy, we define the similarity of two situations by defining first a vector representation of a situation, and then comparing these vector representations. The semantic vector representation of a contextual situation is defined as the average of the semantic vectors representing its known conditions:

$$\mathrm{w}_s = \frac{1}{|s|} \sum_{c \in s} \mathrm{w}_c \tag{5.8}$$

Then, the similarity between a target situation s^* and a candidate situation s is estimated as the cosine of the angle between their corresponding semantic vectors:

$$sim(s, s^*) = \frac{w_s^T w_{s^*}}{\sqrt{\sum_{i=1}^{l} w_{si}^2} \times \sqrt{\sum_{i=1}^{l} w_{s^*i}^2}} \tag{5.9}$$

Figure 5.3 shows the semantic vectors of three possible contextual situations defined by the conjunctions of two conditions (which are also shown in **Figure 5.2**).



Situation	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
S_1=<sunny, family>	0.5	-1	0.6	0.5	0.5	-0.4
S_2=<family, happy>	0.4	-0.9	1.1	0.6	0.9	-0.4
S_3=<sunny, happy>	0.7	-0.8	0.7	1	1	-0.8

Figure 5.3. Situation-to-item influence matrix – semantic vectors

5.3.3. *Exploiting the Relevance of Contextual Conditions*

An important feature of contextual information, which was not mentioned before, is that some contextual conditions may have a larger influence on the ratings than others, and these differences might be exploited to define a more effective situation-to-situation similarity. In fact, the contextual factors that do not have a significant impact on the ratings are not useful when computing rating predictions: they are noisy information. For that reason, recent works have proposed detection methods for deciding whether a specific contextual factor should be captured and exploited during the recommendation process or not. Some of them are based on user surveys [Baltrunas et al. 2012], i.e. directly asking the users about their perceived importance of the contextual factors in their decision processes. Other methods are based on the offline analysis of the rating data by using significance testing methods. For instance, Odić et al. [2013] proposed a method to identify the relevant contextual factors using Pearson's chi-squared. Each of these techniques has its advantages and drawbacks: a survey assessment may be better in detecting the effect of context on the final acceptance of recommendation, but requires more user-effort and may identify factors that are not ultimately correlated to the ratings in the training data set, while offline methods are better in finding quantitative correlations between contextual conditions and ratings, but may be unreliable when the data are sparse.

Similarly to Odić et al. [2013], we proposed a data-driven method to quantify the relevance of contextual information, but in our case at condition level rather than factor one. Taking advantage of our method to measure the influence of conditions (with either respect to items or users), presented in Section 5.3.1, and assuming that the larger the influence of a condition, the more relevant is, we define the relevance of contextual conditions as the *variance* of their corresponding semantic vectors. Denoting by r_c the relevance of a condition c, by w_c its influence vector with respect to items or users, and by μ_c its mean influence (i.e. $\mu_c = \frac{1}{l}\sum_{i=0}^{l} w_{ci}$), we have that:

$$r_c = \frac{1}{l}\sum_{i=0}^{l} (w_{ci} - \mu_c)^2 \tag{5.10}$$

We propose two different ways for using the above contextual conditions' relevance definition when building the semantic vectors:

- a *weighting* method, in which the relevance is used to weight the contribution of each individual semantic vector when computing the representation of a contextual situation defined by several conditions;
- a *filtering* method, where the semantic vectors of the least relevant conditions are excluded from the definition of a situation's vector (i.e. their relevance weight is set to 0).

The *weighting* method consists of modifying the semantic representation of a contextual situation (see Eq. 5.8) by using a weighted average of the semantic vectors of its composing conditions, where the weights are the relevancies of each known condition. Therefore, this method can only be applied with the *direct* strategy to estimate situation-to-situation similarities. Incorporating the definition of relevance in Eq. 5.8, the new formula to compute the semantic vector of a situation s is the following one:

$$\mathrm{w}_s = \frac{\sum_{c \in s} \mathrm{w}_c \cdot r_c}{\sum_{c \in s} r_c} \tag{5.11}$$

The *filtering* method consists of excluding the q least relevant conditions when computing the semantic representations. Differently from the previous case, this method can be also applied with the *best-pairs* and *all-pairs* similarity strategies, since it completely removes the contribution of the semantic vectors of the excluded conditions. When computing the similarity between two conditions (Eq. 5.5), if c or c' is one of the excluded conditions, then $sim(c, c') = 0$; and when computing the semantic vector of a situation (Eq. 5.8) the weight associated to the semantic vectors of the non-relevant conditions is set to 0. Additionally, we remove the rows corresponding to the non-relevant conditions before applying SVD to the co-occurrence matrix in order to remove their contribution to the resulting latent semantic representation. (See Section 5.4.3 for more details about the excluded conditions (q) for each data set.)

5.3.4. *Improving Scalability by Using Clustering Techniques*

A potentially severe limitation of SPF, which is common to all the reduction-based approaches, is the need to learn a local rating prediction model for each target contextual situation that the system may face. This means that, depending on the number of possible situations and the size of each local prediction model, SPF can be more memory-demanding than CARS techniques where a unique global prediction model is needed (e.g. contextual post-filtering and contextual modeling approaches). In the worst case scenario, when the complexity of each local model is almost the same as the global model (i.e. the local models have the same number of parameters as the global one) the memory consumption may be really high on large data sets. However, as we will show in Section 5.4.6, the local models that SPF commonly builds are usually trained with a small portion of the training data (i.e. only with those ratings tagged with situations strongly similar to the target one) and thus their complexity in terms of number of parameters is usually lower than global context-aware approaches, such as TF and CAMF.

Moreover, during the initial system test we conjectured that many of the local models produced by SPF may be highly similar to each other. Therefore, “merging” together models trained on almost the same ratings may be an effective approach to save space without significantly sacrificing prediction accuracy. In a set of experiments illustrated in Section 5.4.4, we have studied the effectiveness of some clustering techniques to identify which local models may be merged. In this way, several local models, in the same cluster, can be replaced with a clustered model built by using all the rating data used by the local models belonging to the same cluster.

We have developed and evaluated two different clustering strategies with the combined goal of reducing the number of local models while avoiding any decrease in the prediction accuracy – compared with the original, not-clustered SPF method:

- *K-means--based clustering strategy*. This (models' reduction) strategy uses the K-means clustering algorithm [Rajaraman and Ullman 2012] to find, first, *k* optimal groups of contextual situations and then for all the situations in a cluster (and for each cluster) it selects, as relevant set of training ratings, those acquired in the situations associated in the cluster.
- *Hierarchical clustering strategy*. This strategy uses a bottom-up hierarchical clustering which begins by considering each local set of ratings (i.e. the ratings associated to each target situation) as a cluster, and then gradually, moving up the hierarchy, merges the most similar pairs of clusters.

Note that each clustering strategy employs a different similarity function for the clustering: the *k-means*--based strategy estimates situation-to-situation similarities by directly comparing their semantic vectors, whereas the *hierarchical* strategy compares two situations based on the similarity of sets of ratings previously selected as relevant for the given situations.

***k-means*--based clustering strategy.** We implemented the *k-means*--based strategy using the standard algorithm (also known as Lloyd's algorithm). **Figure 5.4** shows the pseudo-code of the full learning process using the standard k-means algorithm.

```
Input
  P: initial set of contextual situations to be clustered
  k: exact number of clusters to produce
Output
  models: k local prediction models (one for each cluster)
  clusters = getRandomlyInitialGroups(P, k);
  movement = true;
  while movement do
    movement = false;
    centroids = computeMeans(clusters);
    foreach $p \in P$ do
      nearestClusterID = findNearestCluster(p, centroids);
      if ($p.clusterID \neq nearestClusterID$) then
        assignPointToNearestCluster(p, clusters, nearestClusterID);
        movement = true;
      endif
    endfor
  endwhile
  foreach c ∈ clusters do
      $R_c$ = getRelevantRatings(c);
      $M_c$ = buildLocalModel($R_c$);
      addModelTo(models, $M_c$);
  endfor
return models
```

Figure 5.4. Algorithm of the full learning process using the k-means--based clustering strategy

The process has two parameters as input: the set of contextual situations to be clustered, and the exact number of clusters to produce. The algorithm begins initializing the clusters by randomly assigning each semantic vector of a situation to one of the possible *k* clusters. Then, it follows an iterative process, which ends when the clustering converges, that is, when all the situations are assigned to the nearest cluster. In each iteration, two main operations are carried out:

1. The centroid of each cluster is updated computing the average situation (i.e. semantic vector) over the set of situations currently associated to the cluster.
2. Each situation is moved to the nearest cluster, based on the Euclidean distance between the centroid of the cluster and the semantic vector of the situation.

Note that here we use the Euclidean distance rather than the proposed cosine situation-to-situation similarity (Eq. 5.9). The reason is because the standard k-means algorithm assumes a Euclidean space, and thus using other distance metrics will not ensure the convergence of the algorithm [Rajaraman and Ullman 2012]. Once the clusters of contextual situations are generated, we build a local predictive model for each of the clusters using as training set only the ratings acquired in situations that belong to the cluster. In this case, instead of using the similarity threshold, the level of expansion of the local models is controlled by the meta-parameter *k*: the larger *k*, the fewer contextual situations are aggregated per cluster. Similarly to the global similarity threshold (*t*), the optimal *k* for each data set must be determined experimentally.

Hierarchical clustering strategy. This strategy builds clusters of contextual situations by directly merging the local sets of ratings associated to each possible situation. We have implemented a bottom-up approach that gradually merges highly similar pairs of ratings sets. In this case, we measure the similarity between ratings sets using the standard *Jaccard* similarity, which calculates an estimation of how well the two sets overlap as follows, being R_{s1} and R_{s2} the ratings sets of two different contextual situations s_1 and s_2 respectively:

$$sim(R_{s1}, R_{s2}) = \frac{R_{s1} \cap R_{s2}}{R_{s1} \cup R_{s2}} \tag{5.12}$$

Figure 5.5 shows the pseudo-code of the full learning process using the proposed hierarchical clustering strategy. It takes as input the local sets of ratings for each possible target contextual situation, and a similarity threshold that determines the minimum *Jaccard* similarity between two sets so that they can be merged. The algorithm follows an iterative process than ends when no more sets can be merged in a new cluster. At each iteration, the most similar pairs of sets are merged if their similarity is larger than the specified threshold. When this happens, the original local sets are removed and the merged one is added as a new set of ratings. The resulting sets of ratings correspond to the new clusters, each of which contains the contextual situations associated with each merged set. As in the previous strategy, the final step consists of building the local prediction models using the ratings associated to each resulting cluster.

```
Input
  ratingSets: Initial sets of ratings (one set for each possible target contextual situation)
  $t_m$: Jaccard similarity threshold
Output
  models: local prediction models (one for each cluster)


  merge = true;
  while merge == true do
    merges = false;
    foreach $R_s \in ratingSets$ do
      $R_{s'}$ = findMostSimilarSet($R_s$, ratingSets);
      if (JaccardSim($R_s$, $R_{s'}$) > $t_m$) then
        $R_c$ = union($R_s$, $R_{s'}$);
        addSetTo(ratingSets, $R_c$) ;
        removeSetFrom(ratingSets, $R_s$);
        removeSetFrom(ratingSets, $R_{s'}$);
        merge = true;
      else
        addSetTo(ratingSets, $R_s$);
    endfor
  endwhile
  foreach $R_c \in$ ratingSets do
      $M_c$ = buildLocalModel($R_c$);
      addModelTo(models, $M_c$);
  endfor

return models
```

Figure 5.5. Algorithm of the full learning process using the bottom-up hierarchical clustering strategy

5.4. Experimental Evaluation

In order to evaluate the prediction accuracy of SPF, we have considered six contextually-tagged data sets of ratings with different characteristics. **Table 5.1** illustrates descriptive statistics of the data sets.

- The *Music* data set contains ratings for music tracks collected by an in-car music recommender developed by Baltrunas et al. [2011a]. This data set has multiple ratings for the same tracks and users but in different contextual situations, which are described by one condition only. Contextual situations are defined by the union of eight different factors, such as *driving style*, *mood* and *landscape*, and each of the factors can have different conditions (e.g. *active*, *passive*, *happy* and *sad* are possible conditions of the *mood* factor).

- The *Tourism* data set contains ratings for POIs in the region of South Tyrol. It was collected using a mobile tourism recommender system, called *South Tyrol Suggest* (STS) and developed at the Free University of Bozen-Bolzano. In this data set, ratings are acquired in contextual situations described by the conjunction of several conditions, expressed using 14 different factors, such as *weather*, *companion* and *travel goal*. Possible *travel goal* conditions are, for instance, *business*, *education*, *fun* and *social event*. In Appendix B, we show the complete list of contextual factors and conditions represented in this data set, which were identified in Baltrunas et al. [2012].
- The *Adom* data set is derived from the movie rating data set used by Adomavicius et al. [2005]. The ratings were collected in a survey of college students who also provided information about the context of the movie-watching experience. In the data set, conditions are expressed using four contextual factors: *companion*, *day of the week*, *movie venue*, and if it was on the *opening weekend*. As in the previous data set, ratings contain situations described by several conditions (e.g. a situation could be defined by *summer*, *home* and *alone* at the same time).
- The *Comoda* movie-rating data set was collected and used by Odić et al. [2013]. As in the previous data set, it contains ratings acquired in situations defined by the conjunction of several conditions, expressed using 12 different contextual factors, such as *mood*, *time of the day*, and *weather*. See Appendix B for a more detailed description of the contextual factors and conditions represented in this data set.
- The *Movie* rating data set corresponds to the extension of the well-known MovieLens data set (which we also used in the offline evaluation presented in Section 4.4). However, in this experimentation, we used the tags provided by the users to the movies as contextual situations rather than as item content data, based on the assumption that user tags provide a contextual clue of why the movie is important for the user. Given the inherent noise of user-generated tags, we only used those tags that have a statistically significant impact on the user's rating behavior and have been used by a minimum of 5 users. As significance test we used Pearson's chi-squared that has been proved an effective method to identify relevant contextual information [Odić et al. 2013]. We selected the tags that are dependent on the ratings (at 99% confidence level). After this tag filtering process, 29 tags remained as relevant enough to be used as contextual conditions. In this case there is a factor for each condition, i.e., a tag can appear or not in the definition of a contextual situation.
- The *Library* book-rating data set was collected from the LibraryThing website[14] and augmented with user tags. Again, here tags are used as contextual conditions. In this case, given the large number of ratings and tags, we used a stricter tag filtering criteria: in addition to tags significantly

[14] See [www.librarything.com], accessed October 27th, 2013.

influencing the ratings (at the 99% confidence level), we only considered tags that were used by a minimum of 200 users. After the filtering process, 149 tags were kept as relevant.

Table 5.1. Data set's statistics

Data set	ratings	rating scale	users	items	density	factors	conditions	conditions per situation	situations
Music	4013	1-5	43	139	16%	8	26	1	26
Tourism	1358	1-5	121	101	15%	14	57	2,79	375
Adom	1464	1-13	84	192	9.1%	4	14	3,20	134
Comoda	2296	1-5	121	1197	1.4%	12	49	11,84	1939
Movie	2190	1-10	428	1115	0.4%	-	29	2,15	114
Library	609K	1-10	7192	37K	0.2%	-	149	3,65	39K

Similarly to previous research on CARS, we evaluated SPF in terms of rating prediction accuracy (MAE) and the improvements produced by SPF with respect to a context-free recommendation technique. We measured the accuracy of SPF by conducting a per-user evaluation protocol known as *all-but-n* because, as noted by Shani and Gunawardana [2011], it is better than the traditional *n-fold* cross-validation in assessing the true improvement as perceived by the users (both users with many and few ratings count equally). Using this protocol, for each user, *n* ratings are randomly selected as test set (we used n=5 in *Library* and n=3 in the other data sets) and all the remaining ratings of the user are used for training. We note that the training ratings were also used to acquire the distributional semantics of contextual conditions (but not the test ratings). Because we wanted to reduce the computational cost of the experiments on the *Library* data set, in this case we restricted the maximum number of possible target contextual situations to 100. Additionally, in all the data sets we only selected as possible target contexts those situations that were present in the training set with at least three ratings. All the reported results are averages of per-user evaluations, and the statistical significance of the differences between evaluated models has been calculated using the paired Wilcoxon sign rank test.

For building the local prediction models we used the MF model proposed by Koren [2010], known as *bias MF*, since it is one of the best-performing rating prediction models, especially when dealing with highly sparse data. The estimation function of *bias MF* is defined in Eq. 2.23 in Section 2.4.3. Nonetheless, note that in SPF it is possible to use any available context-free rating prediction model to build the local models.

The numeric meta-parameters of the evaluated models (i.e. similarity thresholds, regularization parameters, etc.) were optimized on a single validation set by using the Nelder and Mead [1965] simplex search algorithm, which has been proved to be an effective solution for multi-dimensional meta-parameter optimization. This method begins with a set of points, each of which represents a specific configuration and which, together, form the initial working simplex. At each iteration, the method performs a sequence of transformations of the simplex with the goal of finding a new configuration that decreases the MAE on the validation set with respect to the previously evaluated configurations. In our experiments the algorithm converged, on average, after 15 iterations.

We built the validation set by applying the per-user splitting protocol mentioned above on the original training set, and we used the resulting two sets, the new and smaller training set and the new test or validation set, for the meta-parameter optimization. The only difference with respect to the first training-test split is that in this case we restricted the ratings selected for validation to those tagged with contextual situations that appear in the test set. We applied this constrain in order to have the set of target situations in the validation set as similar as possible to the one in the test set. Thus, the meta-parameters were optimized for a different set of ratings but similar target contextual situations. For the local MF models produced by SPF, we employed the same optimal meta-parameters used by the global MF model.

We have considered two baseline algorithms:

- the context-free *bias MF* model (which here we denoted by *MF*), which generates rating predictions without considering the context, i.e., all predictions are based on a global prediction model learnt using all the training ratings; and
- *Exact Pre-filtering*, the reduction-based variant proposed by Adomavicius et al. [2005] (described in 2.5.2) in combination with *bias MF* (which here we denoted by *Pref-MF-Exact*).

5.4.1. *Comparison of the Situation-to-Situation Similarity Measures*

In this section, we evaluate the effectiveness of the three *situation-to-situation* similarity measures described in Section 5.3.2:

- *SPF-MF-BP* corresponds to SPF using the *best-pairs* similarity measure defined in Eq. 5.6.
- *SPF-MF-AP* corresponds to SPF using the *all-pairs* similarity measure defined in Eq. 5.7.
- *SPF-MF-DR* corresponds to SPF using the *direct* similarity measure defined in Eq. 5.9.

Figure 5.6 shows the MAE reduction with respect to the baseline algorithm (*MF*) of the SPF variants mentioned above using the best-performing method for building the semantic vector representation of contextual conditions in each considered data set (see next section for a performance comparison of the proposed semantic vector representations). We can see that the *direct* measure (*SPF-MF-DR*) clearly achieves better results than the other SPF variants (*SPF-MF-AP* and *SPF-MF-BP*). As expected, in the *Music* data set, where situations are defined by one condition and thus only direct condition-to-condition similarities can be used, the results of both similarity measures are exactly the same.

As it can be observed in **Figure 5.6**, the differences among variants are larger in the data sets where the average number of conditions per situation is larger. For instance, in the *Comoda* data set, where contextual situations are defined on average by 12 conditions, *SPF-MF-DR* improves the accuracy by 12% with respect to *SPF-MF-AP* and by 14% with respect to *SPF-MF-BP*, which in this case almost has the same accuracy as *MF*. In the other data sets, where situations are defined on average by 3 conditions, the improvement is not that large (3% on average). Comparing the pairwise strategies, it is also clear that the *all-pairs* strategy is more effective than the *best-pairs* (*Pref-MF-BP).*

Furthermore, *SPF-MF-DR* has a smaller time complexity compared with *Pref-MF-AP*. In the worst case, computing the *all-pairs* aggregation of condition-to-condition similarities for two contextual situations is O(n^2), whereas the cost of computing the averaged semantic vector for two situations is at most O(2*n*); *n* being the number of contextual factors.

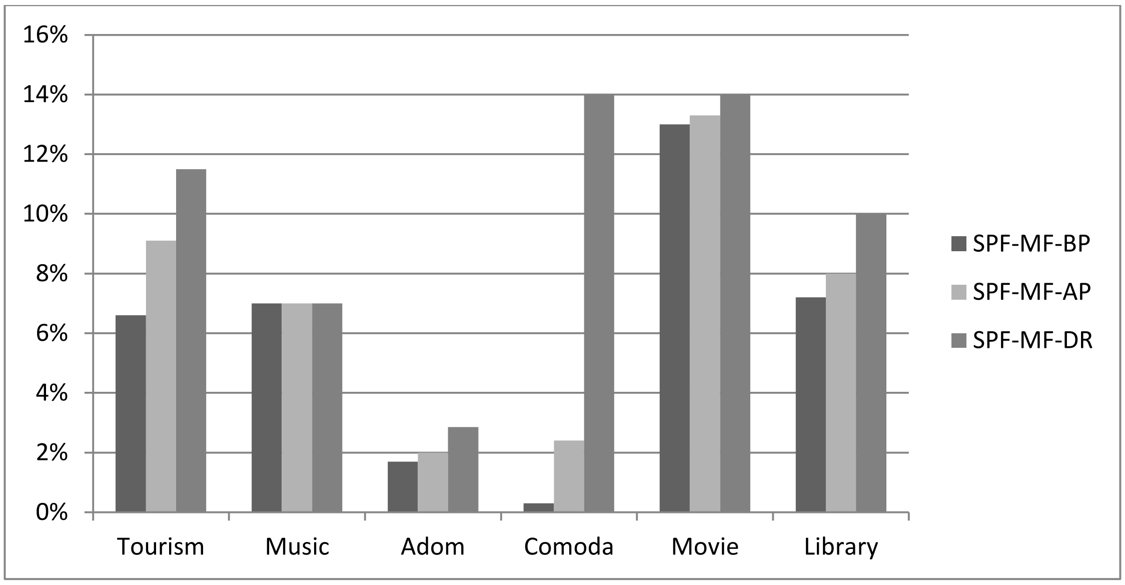


Figure 5.6. MAE reduction with respect to *MF* of SPF variants using different situation-to-situation similarities measures

5.4.2. *Semantic-Vector Representations*

In this section we compare the performance of SPF when different methods (described in Section 5.3.1) are used to obtain the distributional semantics of contextual conditions, i.e., the semantic vectors. To simplify the performance comparison, here we only show the results of SPF using the *direct* situation-to-situation similarity measure, which is the one that performs better according to the results shown in the previous section; however, we obtained similar results when using the other similarity measures. The variants evaluated in this section are the following:

- *SPF-MF-UB*, which uses the user-based perspective for measuring the influence of conditions (defined in Eq. 5.2) and building the semantic vectors.
- *SPF-MF-IB*, using the item-based perspective, defined in Eq. 5.3, to build the semantic vectors.
- *SPF-MF-IC,* which employs the item categories perspective for measuring the influence of conditions, as in Eq. 5.4.

- *SPF-MF-UB-SVD*, which uses SVD to reduce the dimensionality of the semantic vectors built by using the user-based perspective of influence.
- *SPF-MF-IB-SVD*, which applies SVD to the condition-to-item co-occurrence matrix instead.

Table 5.2 shows the MAE of each SPF variant compared to the baseline algorithms *MF* and *Pref-MF-Exact*. As it can be observed, no SPF variant is clearly superior to the others. However, the results show that, when the best variant is used, SPF clearly outperforms the baseline algorithms.

We note that dimensionality reduction techniques are not always improving the results. In fact, in some data sets (e.g. *Comoda* and *Movie*) the accuracy is worse when using them. This can be explained by the fact that SVD is useful only when the matrix to be decomposed has redundant data; otherwise, SVD is also removing/compressing relevant information. In the data sets where SVD is beneficial, we can see that the improvement achieved is relatively small compared to not using it; only in *Music* it is significant. This overall small improvement in all these data sets can be explained by the fact that the original co-occurrence matrices have almost no redundant data, since the available contextual conditions are few and informative. We experimentally optimized to each data set the number of latent factors used in SVD. In particular, we used 20 factors in *Tourism*, 10 factors in *Music* and *Adom*, and 130 factors in *Library*.

Comparing the two reduction techniques, it can be observed that, in these data sets, the variant using semantic vectors based on explicit item categories (*SPF-MF-IC*) is slightly worse than the ones using SVD. Note that *SPF-MF-IC* can only be used in the data sets where items are classified in categories, which in this experimentation are *Music*, *Tourism* and *Comoda*, and its effectiveness strongly depends on how meaningful the categorization of the items is. If contextual conditions do not have a stable influence on the items belonging to the category, then the reduced semantic vectors may be noisy.

Table 5.2. MAE results of SPF using different methods for building the semantic vectors of conditions. Results in bold are statistically significant better (95% confidence level) than the baseline algorithms. Underlined results are also significantly better than the other SPF variants.

Model	Tourism	Music	Adom	Comoda	Movie	Library
MF	1.00	1.00	2.25	.76	1.27	1.26
Pref-MF-Exact	1.02	1.21	2.21	.83	1.13	1.19
SPF-MF-UB	.94	**.94**	2.19	<u>**.65**</u>	1.12	**1.14**
SPF-MF-IB	**.89**	.99	2.21	**.68**	<u>**1.10**</u>	**1.14**
SPF-MF-IC	**.91**	.99	**-**	**.69**	-	-
SPF-MF-UB-SVD	**.90**	.97	2.19	**.69**	1.12	**1.14**
SPF-MF-IB-SVD	**.88**	**.93**	2.19	**.71**	1.12	**1.15**

Figure 5.7 focuses on the performance comparison of the item-based and user-based perspectives that can be used for estimating the influence of contextual conditions. It can be observed that again there is no a clear winner, since item-based perspective performs better in *Tourism*, *Music* and *Movie*, and the user-based one achieves better results in *Adom*, *Comoda* and *Library*. Similarly to other CARS techniques, such as CAMF [Baltrunas et al. 2011] and *User-Item Splitting* [Baltrunas and Ricci 2009; Zheng et al. 2013], different variants are possible depending on whether context is modeled with respect to the users or items, and it is difficult at design stage to decide which variant is better. Furthermore, note that in some data sets, such as *Adom* and *Library*, the differences between perspectives are not significant, which means that not always this decision is crucial for the effectiveness of SPF. Similarly to User-Item Splitting [Zheng et al. 2013], we also evaluated an SPF variant modeling the context with respect to items and users at the same time but it did not improve the results.

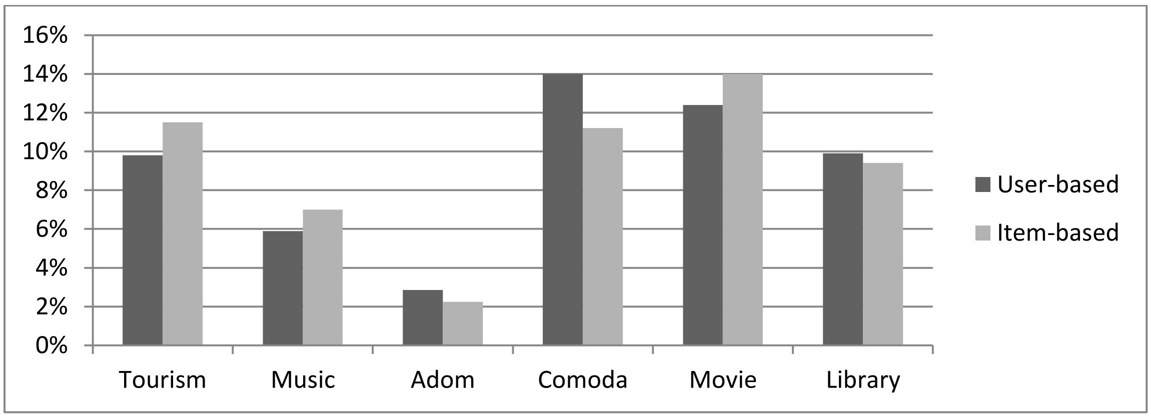


Figure 5.7. MAE reduction of best SPF variant using user-based semantics and item-based semantics with respect to *MF*

Additionally, we have analyzed and compared the condition-to-condition similarities calculated using the proposed item-based and user-based perspectives in order to bring some examples illustrating why they perform differently in different data sets. In **Table 5.3** we show as example the top-10 similar conditions to *cloudy day* and *negative mood* conditions of *Tourism* and *Comoda* data sets, respectively. (A more complete description of the contextual conditions and factors used in both data sets is shown in Appendix B.)

Looking at the top ranked conditions in *Comoda*, one can see that, in general, user-based similarities, which are the best-performing ones, seem to be also more reliable in terms of common sense. For example, in *item-based* the *happy* emotion appears in the top-10 of *negative* mood, which contradicts common sense, whereas in *user-based* the *disgusted* emotion[15] is the most similar one. Analogously, in *Tourism, item-based* similarities (best-performing here) seem to be more reliable in terms of common sense than *user-based* ones (in which, e.g., the top-2 condition of *cloudy day* is *fun*, which could be seen as puzzling association in the tourism domain).

Based on the system performance results and the characteristics of the contextual information captured in the evaluated data sets, we concluded that in data sets like *Comoda*, where user-dependent contextual factors are more predominant and relevant to the domain (e.g. mood, emotions, and physical aspects), using the *user-based* perspective is better; and, in data sets like *Tourism*, where environmental contextual factors are more relevant (e.g. weather, temperature, time) using the *item-based* perspective seems a better option.

Table 5.3. Top-10 similar conditions to *cloudy day* (*Tourism* data set) and to *negative mood* (*Comoda* data set) using user-based and item-based perspectives. In parenthesis we show the factor of each condition.

Tourism (top-10 similar to *Cloudy* day)		**Comoda (top-10 similar to *Negative* mood)**	
User-based	**Item-based**	**User-based**	**Item-based**
Budget traveler(Budget)	Winter(Season)	Disgusted(EndEmotion)	Angry (EndEmotion)
Fun(Goal)	Cold(Temperature)	Disgusted(DominantEmotion)	Not chosen (Decision)
With friends(Companion)	Rainy(Weather)	Afternoon(Time)	Night (Time)
Near by(Distance)	Social-event(Goal)	Neutral (Mood)	First (Interaction)
Working day(DayType)	Snowing(Weather)	Friend's house(Location)	Spring (Season)
Visiting friends(Goal)	Empty(Crowdedness)	Neutral (EndEmotion)	Scared (EndEmotion)
Rainy(Weather)	Night-time(Time)	Neutral (DominantEmotion)	Neutral(DominantEmotion)
Alone(Companion)	Not-crowded(Crowdedness)	Not chosen (Decision)	Happy (EndEmotion)
Returning visitor(Knowledge)	Business(Goal)	First (Interaction)	Afternoon(Time)
Crowded(Crowdedness)	Public-Transport(Transport)	Working day (DayType)	Chosen(Decision)

5.4.3. *Context Relevance*

In this section we evaluate the effectiveness of the methods that we have introduced for exploiting our definition of conditions' relevance (described in Section 5.3.3). **Figure 5.8** shows the MAE reduction with respect to *MF* of the following SPF variants:

- *SPF-MF-DR*, which is a variant of SPF using the *direct* similarity function but not using the relevance weighting;
- *SPF-MF-DR-Weighted* is the method using the weighted average semantic vector calculation defined in Eq. 5.11;

[15] *EndEmotion* factor refers to the user's emotional state at the end of the movie-watching experience and *DominantEmotion* factor refers to the emotional state during the movie-watching experience.

- *SPF-MF-DR-Filtered* is the method that directly excludes the *q* least relevant contextual conditions during the calculation of the situation-to-situation similarities.

As shown in **Figure 5.8**, there is no clear winner: *SPF-MF-DR-Weighted* yields a lower MAE in *Adom*, *Movie* and *Library* data sets, whereas *SPF-MF-DR-Filtered* is slightly better in *Tourism* and *Comoda*. In general, it is demonstrated that the exploitation of the relevance of conditions is useful to improve the effectiveness of SPF, although in some data sets this improvement is little significant.

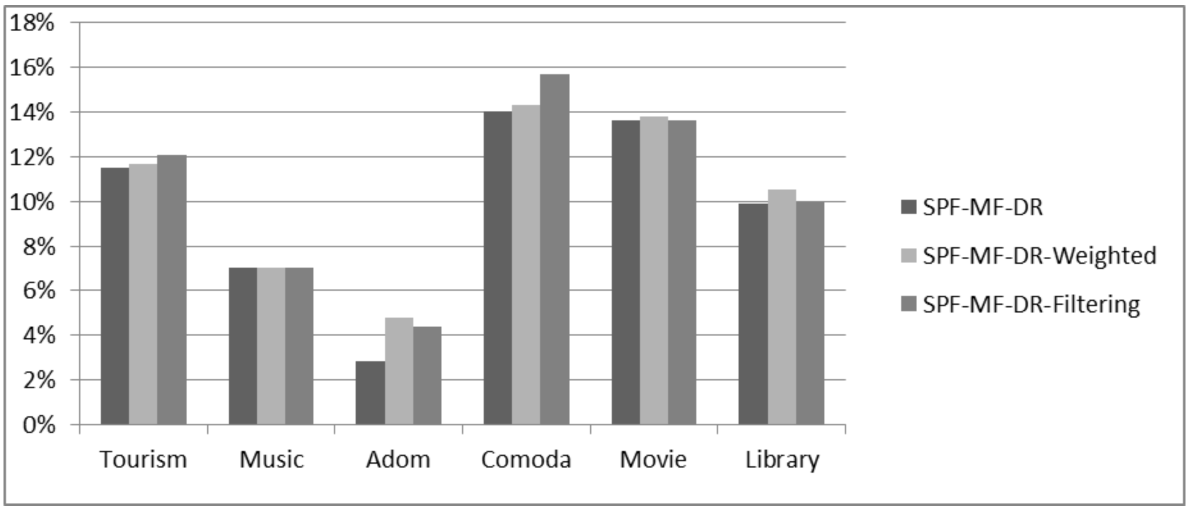


Figure 5.8. MAE reduction of SPF variants using the relevance weighting methods with respect to *MF*

Based on the characteristics of the contextual information in each data set, it seems that *SPF-MF-DR-filtered* performs better in data sets with a relatively high number of conditions and high context granularity. We believe that under these conditions it is more likely that some conditions may introduce noise, and thus it is better to directly exclude them from the model rather than weighting their contribution during the similarity calculation. In *Comoda*, the best results (an improvement of 2% with respect to *SPF-MF-DR*) are obtained when considering the top-29 relevant conditions. In *Tourism*, the best performance is obtained when using the top-40 relevant conditions and the semantic definition of the contextual conditions is based on the item categories (improvement of 1%). Note that, according to the results shown in **Table 5.2**, the best-performing semantic definition in *Tourism* was using the item-based perspective combined with SVD. This means that, once one has excluded the noisy conditions from the model, the reduced semantic vectors based on explicit item categories become more effective.

In contrast, *SPF-MF-DR-Weighted* seems to perform better in the tag-based data sets (*Movie* and *Library*) and in *Adom*: it improves MAE by 2% with respect to *SPF-MF-DR*. We note that this method is more effective when the relevance of the conditions significantly differs between them, since in these cases the weighted average has a major impact on the resulting averaged semantic vectors. This happens for instance in the *Adom* data set where the top-3 conditions are largely more relevant than the others.

5.4.4. *Evaluation of the Clustering Strategies*

This section illustrates and compares the effectiveness of the clustering strategies described in Section 5.3.4. The MAE of the best-performing relevance-boosted SPF variants using the two alternative clustering strategies is shown in **Table 5.4**. In addition to the baseline algorithms, in this table we also show the MAE of the best-performing not-clustered SPF variant, which we denote as *SPF-MF-DR+*. The variants using the clustering strategies are denoted by:

- *SPF-MF-DR+-kmeans*, which is the variant using the *k-means--based* clustering method;
- *SPF-MF-DR+-hierarchical*, the variant using the bottom-up *hierarchical* clustering method.

The goal of the proposed clustering methods is to reduce the number of local models generated by SPF while avoiding any decrease of prediction accuracy with respect to *SPF-MF-DR+*. However, we did not expect an improvement of the accuracy when using the proposed model clustering methods. We can observe that *SPF-MF-DR+-hierarchical* slightly outperforms *SPF-MF-DR+* in some data sets, especially in *Tourism*. This demonstrates that merging similar sets of ratings, i.e., sets that mostly overlap, not only helps to reduce the number of local models, but also improves the prediction accuracy. In contrast, the k-means method (*SPF-MF-DR+-kmeans*) always produces a significant loss of accuracy with respect to *SPF-MF-DR+*.

Table 5.4. MAE of SPF using the proposed clustering strategies

Model	Tourism	Music	Adom	Comoda	Movie	Library
MF	1.00	1.00	2.25	.76	1.27	1.261
Pref-MF-Exact	1.02	1.21	2.21	.83	1.13	1.193
SPF-MF-DR+	**.88**	**.93**	2.14	**<u>.64</u>**	**1.10**	**<u>1.13</u>**
SPF-MF-DR+-kmeans	.92	.96	2.19	.69	**1.10**	**1.15**
SPF-MF-DR+-hierarchical	**.86**	.95	2.14	**<u>.64</u>**	**1.10**	**<u>1.13</u>**

Table 5.5 shows the number of local models produced by SPF with and without the model clustering strategies. It can be observed, as an advantage of *SPF-MF-DR+-kmeans*, that in its optimal configuration it produces a much coarser clustering, that is, fewer local MF models are built. In particular, we found that the best results setting is *k* equal to 2 in *Tourism*, *Music* and *Comoda*, 4 in *Adom*, 5 in *Movie* and 8 in *Library*. Conversely, the optimal configuration of *SPF-MF-DR+-hierarchical* merges highly similar sets of ratings, that is, with *Jaccard* similarity larger than 0.8. Therefore, the hierarchical clustering strategy, based on the merging of similar rating sets is more selective in merging similar models and produces a larger number of local models. Using this strategy, the more significant reduction of the number of local

models is produced in the *Tourism* data (70% reduction with respect to SPF-MF-DR+) and *Comoda* (43% reduction with respect to SPF-MF-DR+).

Table 5.5. Number of local MF models produced by each SPF variant

Model	Tourism	Music	Adom	Comoda	Movie	Library
SPF-MF-DR+	103	26	31	90	18	73
SPF-MF-DR+-kmeans	2	2	4	2	5	8
SPF-MF-DR+-hierarchical	31	23	28	51	12	60

According to these results, *SPF-MF-DR+-kmeans* performs well on the tag-based data sets (i.e. *Movie* and *Library*), i.e., in the trade-off between accuracy and the number of produced local models. On these data sets it is able to find finer-grained clusters and hence the accuracy is not as bad as the one achieved by *SPF-MF-DR+-hierarchical*. On the other data sets, *SPF-MF-DR+-hierarchical* outperforms *SPF-MF-DR+-kmeans* in terms of prediction accuracy. The strategy seems to be more effective when the original local sets are relative small and similar, like for instance in *Tourism*.

To illustrate the SPF performance when model clustering is used we have analyzed the time and space complexity of the SPF variants that are using the clustering strategies in comparison to the not-clustered SPF in the *Tourism* data set. We implemented the algorithms in the Java programming language and executed the experiments on a machine with two 2.4 GHz cores. To speed up the learning process of the considered SPF variants we parallelized the local model learning using multi-threading. In **Table 5.6** we show the execution time and memory consumed by each prediction model. As expected, *SPF-MF-DR+-kmeans* with *k* equal to 2 is clearly more efficient than *SPF-MF-DR+* and *SPF-MF-DR+-hierarchical*. In this particular case it is even faster than *MF* (probably because of the parallel processing and the smaller size of the local training sets of ratings) but *SPF-MF-DR+-kmeans* uses slightly more execution memory. The memory usage is similar because, although they use fewer training ratings, the resulting local MF models have a number of parameters similar to the global one. Regarding the complexity of *SPF-MF-DR+-hierarchical*, we can observe that the hierarchical clustering method consumes more memory than k-means (more local models are generated), but still this is less than what it is used by *SPF-MF-DR+* (84% less) and it is twice as fast as *SPF-MF-DR+*.

Table 5.6. Execution time and memory of the SPF variants and MF in the *Tourism* data set

Model	Time (seconds)	Memory (MB)
MF	5	10
SPF-MF-DR+	57	173
SPF-MF-DR+-kmeans	2	16
SPF-MF-DR+-hierarchical	25	28

5.4.5. *Comparing SPF to the State of the Art*

In this section we compare the performance of the best-performing SPF variant in each data set, which we denote here as *SPF-MF*, with two state-of-the-art CARS techniques. In particular, we compare the rating prediction accuracy of *SPF-MF* to CAMF, the contextual modeling approach proposed by Baltrunas et al. [2011b; 2012], and *UI-Splitting*, a novel splitting-based pre-filtering method proposed by Zheng et al. [2013a] (both methods are described in more detail in Section 2.5). CAMF was proved to outperform TF approaches, especially in small-medium--size rating data sets like *Adom* [Baltrunas et al. 2011b], and *UI-Splitting* was proved to outperform CAMF on the *Comoda* data set [Zheng et al. 2013a].

In **Figure 5.9** we show the performance (MAE reduction) of the three evaluated rating prediction models in comparison with *MF*. For each prediction model its best-performing variant, in each data set, is shown. For *SPF-MF* the best results are obtained using *SPF-MF-DR+-hierarchical*, which combines the best relevance-boosted methods with the hierarchical clustering strategy. Concretely, it uses the *filtering* method in *Tourism* and *Comoda*, and the *weighted* method in the others. Moreover, *SPF-MF* uses the best-performing semantic-vector representations on each data set, which are the following: Item Categories (IC) in *Tourism*, Item-Based (IB) in *Music*, *Adom* and *Movie*, and User-Based (UB) in *Comoda* and *Library*. For *CAMF* the best variants are the following: *CAMF-CC* in *Tourism* and *Music*, *CAMF-CI* in *Adom* and *Comoda*, and *CAMF-CU* in *Movie* and *Library*. Finally, similarly to the experimental evaluation presented by Zheng et al. [2013a], for *UI-Splitting* the best results are obtained when using the chi-square significance test at 95% confidence level as splitting criteria, and *bias MF* as context-free prediction model.

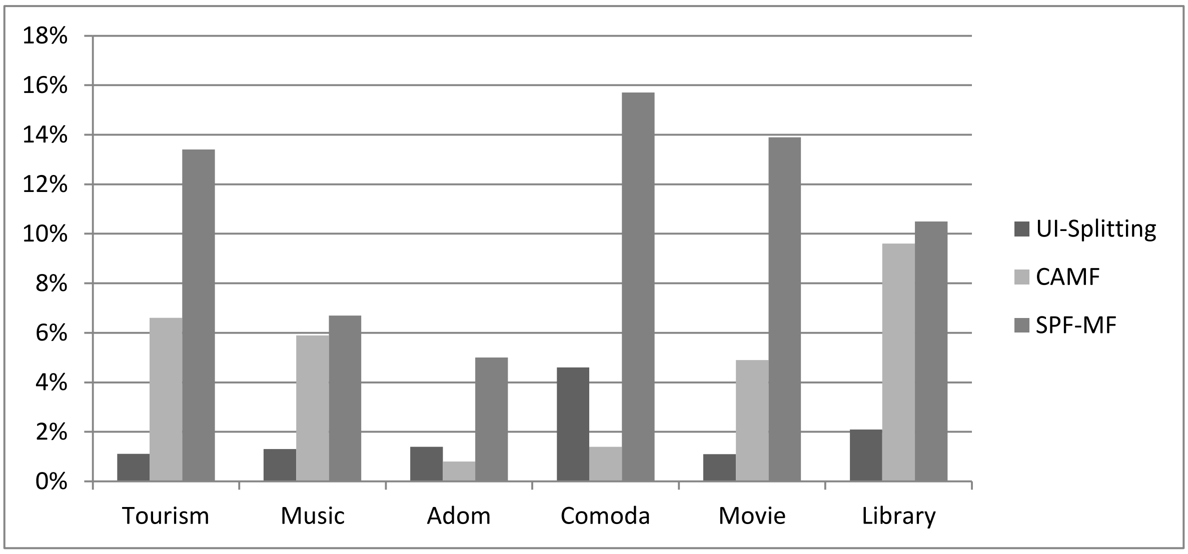


Figure 5.9. MAE reduction with respect to MF (the context-free baseline) of SPF-MF compared to two state-of-the-art context-aware approaches (CAMF and UI-Splitting)

As it can be observed, the three context-aware prediction models significantly outperform *MF* in all the data sets, confirming that contextual predictions are significantly more accurate when context matters. It can be observed that *UI-Splitting* outperforms *CAMF* in *Adom* and *Comoda* data sets, but in the remaining data sets *CAMF* is clearly superior. On the other hand, *SPF-MF* clearly outperforms *CAMF* and *UI-Splitting*, especially in the data sets where the contextual situations are defined by the conjunction of several conditions, like in *Comoda* that has 12 conditions per situation on average, and the improvement achieved with respect to *CAMF* and *UI-Splitting* is 14% and 10%, respectively. In the remaining data sets, where the situations are defined by fewer conditions (between 2 and 4), *SPF-MF* outperforms *CAMF* by 7% in Tourism, 9% in *Movie*, 4% in *Adom*, 1% in *Music* and *Library*. These large differences in accuracy in the data sets where the target contextual situations have finer granularity demonstrate that *SPF-MF* builds better context models than *CAMF* and *UI-Splitting* under those conditions.

5.4.6. *Optimal Level of Contextual Expansion*

A key parameter of SPF is the global similarity threshold that limits the level of contextual expansion with respect to a strict local model. We recall that in SPF, when a rating prediction is requested for a target contextual situation, instead of using exactly the ratings acquired in that situation, we expand the training data by reusing ratings acquired in similar situations, and we build, with these ratings, the local predictive models for the given target contextual situation. In the previous sections, we have illustrated the performance of several variants of SPF; but all of them were using the optimal threshold for each data set, i.e., the threshold that yielded better prediction accuracy.

We note that the impact of the similarity threshold choice on the prediction accuracy is similar to the effect of the *user-to-user* similarity threshold in user-based CF (described in Section 2.4.1): the lower the threshold value, the larger the user's neighborhood. In user-based CF, as in our case, it is important to find the optimal level of locality (neighborhood size), which is the one that cause the system to obtain the best prediction accuracy for the given data.

Table 5.7 shows the level of contextual expansion of the local models produced by SPF using the optimal similarity threshold. We measure the level of contextual expansion applied by SPF based on the amount of ratings added (on average for all the target situations) to the ratings used for building the strict local models targeted to a specific contextual situation (as those built by *Exact Pre-filtering*). Therefore, we say that there is a 0 expansion when no additional ratings are aggregated (i.e. when SPF is using *Exact Pre-filtering*) and 100% expansion when all the ratings in the data set are aggregated independently of the target situation (i.e. when using a context-free global model). Concretely, for a given target situation (s), we measure the expansion level (l_s) applied by SPF to learn its local model as follows: (note that the final percentage is estimated by averaging the l_s over all the tested target situations)

$$l_s = \frac{\#ratings(SPF) - \#ratings(Exact)}{\#ratings(Global) - \#ratings(Exact)} \qquad (5.13)$$

As it can be observed in **Table 5.7**, the precise value of the similarity threshold itself is not providing an indication of the level of contextual expansion applied to a data set, because the expansion depends on the distribution of the *situation-to-situation* similarities. In fact, if the average similarity between situations is small, then a high threshold is more restrictive than if the average is large, since the number of situations that can be aggregated is much smaller. For example, although the optimal threshold in *Adom* and *Comoda* is equal (0.9), the resulting contextual expansion level is much lower in *Adom* than in *Comoda*. The reason is that in *Adom* the average situation-to-situation similarity is 0.38, whereas it is 0.86 in *Comoda*. The same situation is found in *Movie* and *Library*, which have similar levels of contextualization but significantly different optimal thresholds.

Table 5.7. Optimal level of contextualization for each data set using *SPF*

	Tourism	Music	Adom	Comoda	Movie	Library
Similarity threshold	0.3	0	0.9	0.9	0.05	0.35
Contextual expansion	31%	90%	2%	40%	10%	8%

Looking at the results shown in **Table 5.7** side by side to the accuracy data of SPF (shown in **Figure 5.9**), one can note that the improvement of SPF with respect to the global MF model (11% gain in *Library,* 13% in *Tourism,* 14% *Movie* and 16% in *Comoda*) is larger when using a low-medium level of contextual expansion (8% expansion in *Library,* 31% in *Tourism*, 10% *Movie* and 40% in *Comoda*). This fact is not observed for *Adom* data where the improvement is similar to *Music* (5% and 7%, respectively), but *Adom* has a low optimal expansion (2%) while *Music* has a large optimal expansion (90%).

In order to illustrate the effect of varying the similarity threshold on rating accuracy and contextual expansion, in **Figure 5.10,** we show the MAE and the expansion percentage as functions of the similarity threshold in the three movie-rating data sets: *Movie*, *Comoda* and *Adom*. In general, we have noted (not shown in this figure) that using a negative similarity threshold yields bad results. For this reason here we only show the MAE and expansion percentage for positive values of the similarity threshold (from 0 to 1). The charts on the top show the results in the *Movie* data set. As it can be observed, in this case the smallest MAE (i.e. the global minimum) is obtained when the threshold is near to 0.05, which causes a level of contextual expansion above 10%. In *Comoda* (charts on the middle), we found that the best MAE is obtained when the similarity threshold is close to 0.9, which implies a level of contextual expansion around 40%. In this case the accuracy suddenly increases when the threshold is over 0.75, and then drastically drops when the expansion percentage is lower than 30%. Finally, the charts at the bottom show the results obtained in *Adom*. Similarly to *Comoda*, the best MAE is obtained when the threshold is set to 0.9, which corresponds to a very low contextual expansion (2%). Note that the expansion decreases linearly as a function of the threshold.

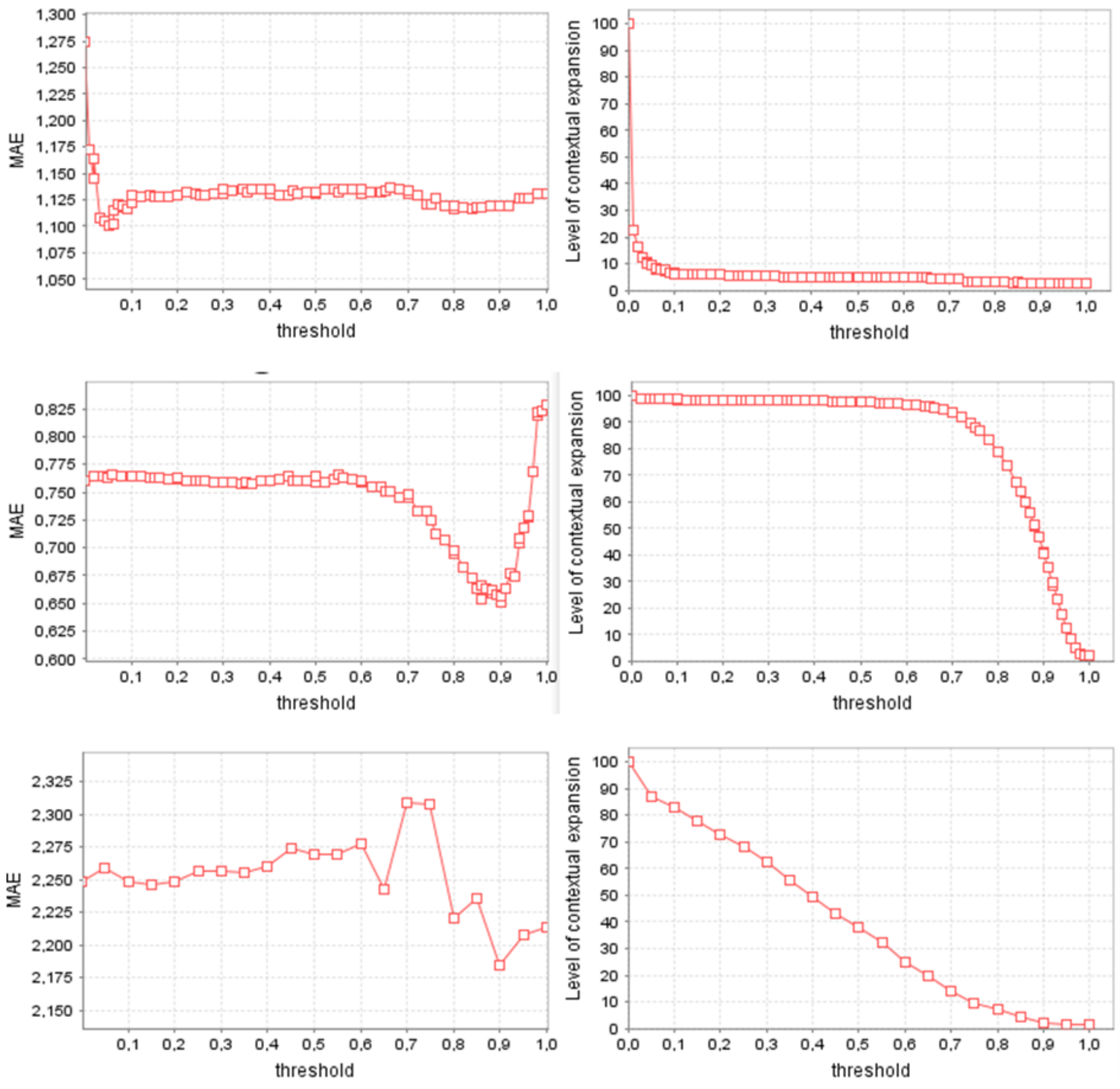


Figure 5.10. MAE (left line chart) and contextual expansion (right line chart) as functions of the similarity threshold (from top to down we show the results in *Movie, Comoda* and *Adom* data sets)

5.5. Conclusions

In this chapter, we have described *Semantic Pre-Filtering* (SPF), a novel SERS technique that is able to tackle the *data-sparsity* problem and thus improve prediction accuracy by exploiting distributional semantic similarities between contextual situations during local context modeling. SPF is a reduction-based approach that employs a global situation-to-situation similarity threshold to precisely select the right level of contextual expansion for a given data set: the larger the threshold, the fewer contextual situations are aggregated to build the local predictive models.

Accurate similarity assessments allow for a fine-grained local context modeling, and the key component of SPF is the method for determining the degree of similarity between a candidate and the target contextual situation. In this chapter, we have introduced our notion of situation-to-situation similarity, which is based on the distributional semantics of contextual conditions, i.e., assuming that two situations are similar if their known conditions influence users' ratings in a similar way. Although the reliability of distributional similarities mostly depends on the size of the training data, it has the advantage that does not require a context taxonomy, as in *Generalized Pre-filtering*.

The experimental evaluation that we have carried out on six contextually-tagged data sets shows that SPF clearly outperforms state-of-the-art CARS techniques when used in combination with *bias MF*. The results show that our approach obtains even better results in data sets where contextual situations have a finer granularity and high data sparsity, demonstrating that SPF is especially effective under these conditions.

Chapter 6 - Developed Recommendation Framework

6.1. Introduction

In the last decade, the use of free, open source software frameworks like *Weka*[16] [Witten et al. 2011] have become a common practice in the machine learning and data mining fields, because they help researchers to implement and evaluate their algorithms by providing stable implementations of existing algorithms and standard evaluation protocols.

More recently, recommendation software frameworks containing several recommendation algorithms and additional functionality to support their evaluation have become also popular in the RS field. Currently, the more complete, free recommendation frameworks are the following:

- *MyMediaLite*[17] [Gantner et al. 2011] is a library written in C#, for the .NET platform, which provides several state-of-the-art CF recommendation algorithms and also some *content-boosted CF* hybrid techniques (see Section 2.6.1 for details about such hybrid strategies);
- *Apache Mahout*[18] [Owen et al. 2011] is a general purpose machine learning library, written in Java, providing several mostly distributed (via *Hadoop*[19]) implementations of recommendation, clustering and classification algorithms. The section dedicated to recommendation algorithms is mainly based on the original *Taste* library[20], and only contains CF recommendation techniques and other baseline algorithms.
- *GraphLab*[21] [Low et al. 2010] is a library written in C++ that contains several implementations of MF recommendation models, such as *SVD++* and *bias MF*.
- *LensKit*[22] [Ekstrand et al. 2011] is a library written in Java that currently contains several CF techniques such as MF prediction models as well as user-based and item-based CF.

Given the lack of support for CB and CARS algorithms of existing recommendation frameworks, in this thesis we implemented our own implementations of the proposed and evaluated recommendation algorithms. But, in order to make our experiments more reproducible by other researchers and

[16] See [http://www.cs.waikato.ac.nz/ml/weka/], accessed on November 14th, 2013.

[17] See [http://www.mymedialite.net/], accessed on November 14th, 2013.

[18] See [http://mahout.apache.org/], accessed on November 14th, 2013.

[19] See [http://hadoop.apache.org/], accessed on November 14th, 2013.

[20] See [http://taste.sourceforge.net/], accessed on November 14th, 2013.

[21] See [http://select.cs.cmu.edu/code/graphlab/index.html] , accessed on November 14th, 2013.

[22] See [http://lenskit.grouplens.org/], accessed on November 14th, 2013.

practitioners, we decided to implement them by extending one of the existing recommendation frameworks. In particular, we extended *Apache Mahout* because at the time of starting this thesis this was the only free, open-source Java-based library, which makes feasible its use with any platform that can run a modern Java Virtual Machine. As in other Apache projects, *Apache Mahout* is built around Maven[23], which is a tool that manages compiling code, packaging release, generating documentation, and publishing formal releases. Detailed instructions for the installation of *Apache Mahout* can be found on its homepage.

This chapter is organized as follows. Section 6.2 describes the main functionality that we have included in *Apache Mahout*. Section 6.3 illustrates how to use the tool that we developed to simplify the offline evaluation of the available recommendation algorithms in *Apache Mahout*. Section 6.4 describes how to use the developed recommendation framework from an existing recommender system. Finally, Section 6.5 summarizes the work presented in this chapter.

6.2. Added Functionality

We have implemented all the proposed recommendation algorithms as well as other state-of-the-art algorithms (included in the previously described experimental evaluations) as part of *Apache Mahout*. For the development of the new recommendation functionality we followed as much as possible the original *Apache Mahout* code structure, which consisted of the following main packages with the "*org.apache.mahout.cf.taste*" namespace:

- **common**: specific data structures implementations and other utility code which are more efficient than normal Java Collections;
- **eval**: code with evaluation protocols and performance metrics;
- **model**: data structures for storing user data;
- **recommender**: CF recommendation algorithms;
- **similarity:** Similarity measures (used by user-based and item-based CF);

6.2.1. *Support for Linear CB Recommendation Algorithms.*

We included classes and interfaces necessary for implementing CB recommendation algorithms. On the one hand, we implemented data structures for representing and accessing item content data. We created a sub-package of *model* named *contentbased* containing these implementations and also different weighting mechanisms. We created the new package *usermodeling* that contains several linear user-profile learning methods, such as the ones proposed by Sen et al. [2009] and the ones proposed in Chapter 4. In the

[23] See [http://maven.apache.org], accessed on November 14th, 2013.

recommender package, we added implementations of the linear CB recommendation algorithms evaluated in this thesis based on the dot product of item and user profiles.

6.2.2. *Support for SGD-based Prediction Models.*

We created a subpackage of *recommenders*, called *sgd*, which contains implementations of model-based recommendation algorithms whose parameters are learnt by using the SGD implementation popularized by Funk [2006], allowing also time-aware rating data explorations. In order to speed up the learning phase of SGD-based models, we also included in the *model* package a more efficient data structure especially optimized for matrix and vector operations.

We implemented several CF and CB recommendation techniques optimized by SGD, some of them also included in the experimental evaluation of the semantically-enhanced recommendation approaches presented in Chapters 4 and 5 (e.g. *bias MF* and *SVD++* [Koren and Bell 2011], the *content-boosted MF* proposed by Forbes and Zhu [2011], and *SGD-CB*). We also included an implementation of BPR-MF [Rendle et al. 2009] ported from *MyMediaLite* as well as another variants also optimized for ranking.

6.2.3. *Support for CARS Techniques*

We created a subpackage of *model*, called *contextaware*, which included data structures for representing and accessing contextual information associated to user ratings. We also provided support for that multi-rating data models, i.e., assuming that in context-aware applications a user can provide several ratings for an item but in different contexts. For instance, the *Tourism* and *Music* data sets used in the evaluation of SPF have this characteristic (see Section 5.4).

We created a subpackage of *recommender*, called *contextaware*, including the implementation of several state-of-the-art CARS techniques (e.g. *timeSVD++* [Koren 2010], *Exact Pre-filtering* [Adomavicius et al. 2005], CAMF [Baltrunas et al. 2011], TF [Karatzoglou et al. 2010], and *User-Item splitting* [Zheng et al. 2013]).

6.2.4. *Support for SERS Techniques*

We created a new package, called *semantics*, containing data structures for representing and accessing ontologies using the Apache Jena library[24]. We implemented several ontology-based and distributional similarity measures (most of them used in the experimental evaluations described in Chapters 4 and 5). In this package we also included implementations of the semantics exploitation methods presented in this thesis, such as the pairwise profile matching strategies and the CSA strategy for user profile expansion.

[24] See [http://jena.apache.org/documentation/ontology/], accessed on November 14th, 2013.

We included implementations of the described SCB and SPF variants, presented in Chapters 4 and 5, respectively, as well as some state-of-the-art SERS techniques like CB-LSA [Bambini et al. 2011].

6.2.5. *Support for Offline Evaluation*

We included an evaluation tool that allows other researcher to test all the available recommendation algorithms in *Apache Mahout* on data provided in text files and without having to write additional code. In *eval* package we included additional evaluation metrics for measuring ranking performance (e.g. AUC, Mean Average Precision (MAP), NDCG, and metrics for measuring novelty and diversity proposed by Vargas et al. [2011]). We also implemented protocols for splitting data in training, validation and test sets, such as the *per-user* chronological splits suggested by Shani and Gunawardana [2011], and the *one-plus-random* protocol proposed by Cremonesi et al. [2010]. In order to speed up the testing phase we included a parallel implementation (via Java Multi-threading) of the evaluation protocols for both rating prediction and ranking recommendation tasks.

Additionally, we have also included a parallel implementation of the popular simplex search algorithm known as Nelder and Mead method [1965], which can be used to obtain the optimal meta-parameter configuration of a recommendation algorithm according to a given optimization criteria, such as RMSE or MAE.

6.3. Using the Evaluation Tool

We have developed a command-line tool to ease the evaluation of the available recommendation algorithms in *Apache Mahout* without having to develop any Java program. The usage of the tool is similar to command-line tools available in other recommendation frameworks, like the one provided by *MyMediaLite*, in which user and item data are provided through files. **Table 6.1** shows the main configurable options to use the evaluation tool, called *recommender_eval*.

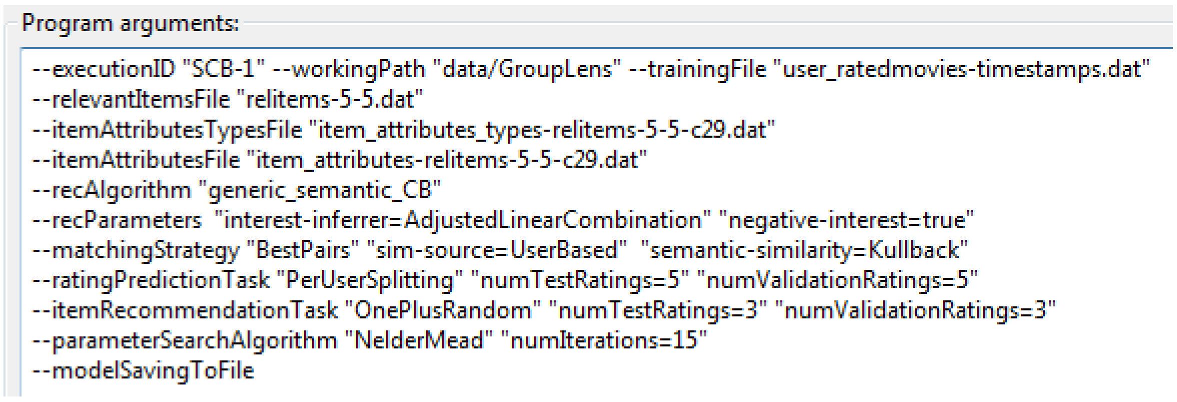
Table 6.1. Configurable options of the command-line evaluation tool

General options:	
--executionID "String"	Define ID of the execution of the recommendation algorithm (used for identifying the generated result files)
--workingPath DIRECTORY	Define DIRECTORY where data and model files are located
--trainingFile FILE	Read user data from FILE. In the case of ratings without context information, each line of FILE contains [userID itemID, rating value] with tab-separated columns. When ratings have context, then the format is [userID, itemID, value, factor1,..., factorN]. The first line is used to define the format of contextual situation, i.e. the order of factors.

--relevantItemsFile FILE	Read list of item IDs that are used for recommendation from FILE. This is used to constrain the original set of items available on user data FILE
--recAlgorithm METHOD	Use METHOD as recommendation algorithm. Some examples are *random* algorithm, *sgd_MF* (which corresponds to *bias MF* using SGD), and *generic_semantic_CB* (which corresponds to SCB).
--recParameters OPTIONS	Use the specific OPTIONS of the recommender. The format is “option1=value2 option2=value2 …” For example, *sgd_MF* may be used with the following configuration: [num-iter=25 learning-rate=0.01 random-exploration=true max-faults=1]
Options for CB:	
--itemAttributesTypesFile FILE	Read from FILE all the item's attributes and their values. Each line of FILE contains a value and its attribute, i.e. [value attribute] (e.g. "comedy genre")
--itemAttributesFile FILE	Read item data from FILE. Each line should contain the one item's attribute value and its relevance [itemID value relevance]
--attributeTypesRelevancesFile FILE	Read the relevance of each item attribute from FILE.
Options for CARS:	
--contextualFactorsFile FILE	Read contextual information from FILE. Each line should contain a contextual condition and its factor, i.e. [condition factor] (e.g. "summer season")
--contextualPrefiltering METHOD [OPTIONS]	Use METHOD as pre-filtering approach (e.g. Exact, SPF variants) and using the specific OPTIONS (in SPF a possible option is setting a specific similarity threshold (e.g. [sim-threshold=0.4])
--contextualPostfiltering METHOD [OPTIONS]	Use METHOD as post-filtering and using the specific OPTIONS
Options for SERS:	
--domainKnowledgeURIs FILE	Read ontology from FILE
--domainKnowledgeFileFormat RDF|OWL	Define the specific format of the ontology (OWL or RDF)
--matchingStrategy METHOD [OPTIONS]	Use METHOD as item-to-user profile matching strategy using the specific OPTIONS, where one can define the semantic similarity measure to use and the global similarity threshold (e.g. [all-pairs sim-threshold=0.25 sem-similarity=])
--profileExpansion METHOD [OPTIONS]	Use METHOD as user-profile expansion strategy (by default CSA) and with the specific OPTIONS.
--contextualMatchingStrategy METHOD [OPTIONS]	Use METHOD as situation-to-situation similarity function (e.g. all-pairs, or direct) with the specific

	OPTIONS.
Evaluation options:	
--ratingPredictionTask METHOD [OPTIONS]	Use METHOD as rating prediction evaluation protocol (by default per-user splitting). Possible OPTIONS are: *numValidationRatings*, *numTestRatings*.
--itemRecommendationTask METHOD [OPTIONS]	Use METHOD as ranking evaluation protocol (by default one plus random protocol). Possible OPTIONS are: *numRandomItems*, *relevantThreshold*.
--parameterSearchAlgorithm METHOD [OPTIONS]	Use METHOD as method for meta-parameter optimization (by default Nelder and Mead method). Possible OPTIONS are the *numberIterations*.
File options:	
--saveModelToFile	Save recommendation model into an automatically generated file (via Java serialization). If the option is enabled, and the file associated to a recommendation algorithm exists, then the model is loaded from the file.
--saveResultsToFile	Save the performance results into files (each metric separated into a different file). Then, the stored results can be used for programmatically creating charts or for significance testing.

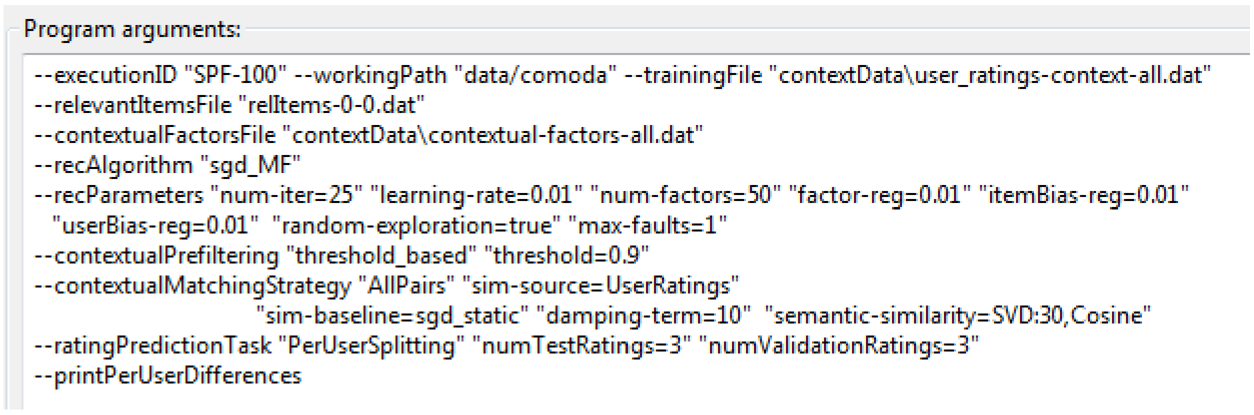
Figure 6.1 illustrates an example of usage in which we show how to evaluate a variant of SCB (the SERS technique described in Chapter 4). "recParameters" define the user-profile learning method used by SCB, which in this example corresponds to the method we have defined in Eq. 4.3 in Section 4.3.1. As profile matching strategy, it uses the *best-pairs* one in combination with user-based distributional similarities (indicated by the option "sim-source=UserBased") that are estimated using the Kullback and Leibler [1951] probabilistic measure (defined in Eq. 3.9). In this case, we evaluate both recommendation tasks: rating prediction and ranking recommendation (denoted by the options "ratingPredictionTask" and "itemRecommendationTask"). For rating prediction evaluation we employ the *per-user splitting* protocol using 5 ratings for test and 5 for validation. For ranking, the *one-plus-random* protocol is used with 3 test and validation ratings per user. Finally, here we also indicate that we want to optimize the numeric meta-parameters of SCB (i.e. the similarity threshold) by using the Nelder and Mead method [1965] with a maximum of 15 iterations.



```
--executionID "SCB-1" --workingPath "data/GroupLens" --trainingFile "user_ratedmovies-timestamps.dat"
--relevantItemsFile "relitems-5-5.dat"
--itemAttributesTypesFile "item_attributes_types-relitems-5-5-c29.dat"
--itemAttributesFile "item_attributes-relitems-5-5-c29.dat"
--recAlgorithm "generic_semantic_CB"
--recParameters  "interest-inferrer=AdjustedLinearCombination" "negative-interest=true"
--matchingStrategy "BestPairs" "sim-source=UserBased"  "semantic-similarity=Kullback"
--ratingPredictionTask "PerUserSplitting" "numTestRatings=5" "numValidationRatings=5"
--itemRecommendationTask "OnePlusRandom" "numTestRatings=3" "numValidationRatings=3"
--parameterSearchAlgorithm "NelderMead" "numIterations=15"
--modelSavingToFile
```

Figure 6.1. Arguments of *recommender_eval* to evaluate a SCB variant.

Figure 6.2 illustrates another example of usage, in this case evaluating a specific SPF variant. Here “recAlgorithm” determines which context-free recommendation algorithm will be used in combination with SPF, and “recParameters” its meta-parameter configuration. In this example, the local models are built by using the SGD-based *bias MF* with the following configuration: 25 iterations over training data using random exploration (i.e. not chronological order), a learning rate of 0.01, 50 latent factors for representing users and items, and setting the factor, user, and item regularization parameters to 0.01. The SPF variant used is indicated by the option “contextualPrefiltering”; in this case “threshold_based” corresponds to the SPF variant not using relevance information and clustering strategies, and with the global similarity threshold set to 0.9. The situation-to-situation similarity measure used is defined by the option “contextualMatchingStrategy”, and it consists of the *all-pairs* strategy, defined in Eq. 5.7 in Section 5.3.2, using the user-based perspective for semantic-vector representation of conditions applying SVD with 30 latent factors, and damping term (β) equal to 10. In this case, SPF is evaluated in terms of rating prediction accuracy using the per-user splitting protocol with 3 ratings per user test and validation.



```
--executionID "SPF-100" --workingPath "data/comoda" --trainingFile "contextData\user_ratings-context-all.dat"
--relevantItemsFile "relItems-0-0.dat"
--contextualFactorsFile "contextData\contextual-factors-all.dat"
--recAlgorithm "sgd_MF"
--recParameters "num-iter=25" "learning-rate=0.01" "num-factors=50" "factor-reg=0.01" "itemBias-reg=0.01"
  "userBias-reg=0.01"  "random-exploration=true" "max-faults=1"
--contextualPrefiltering "threshold_based" "threshold=0.9"
--contextualMatchingStrategy "AllPairs" "sim-source=UserRatings"
                              "sim-baseline=sgd_static" "damping-term=10"  "semantic-similarity=SVD:30,Cosine"
--ratingPredictionTask "PerUserSplitting" "numTestRatings=3" "numValidationRatings=3"
--printPerUserDifferences
```

Figure 6.2. Arguments of *recommender_eval* to evaluate a SPF variant.

6.4. Using the Recommendation Framework

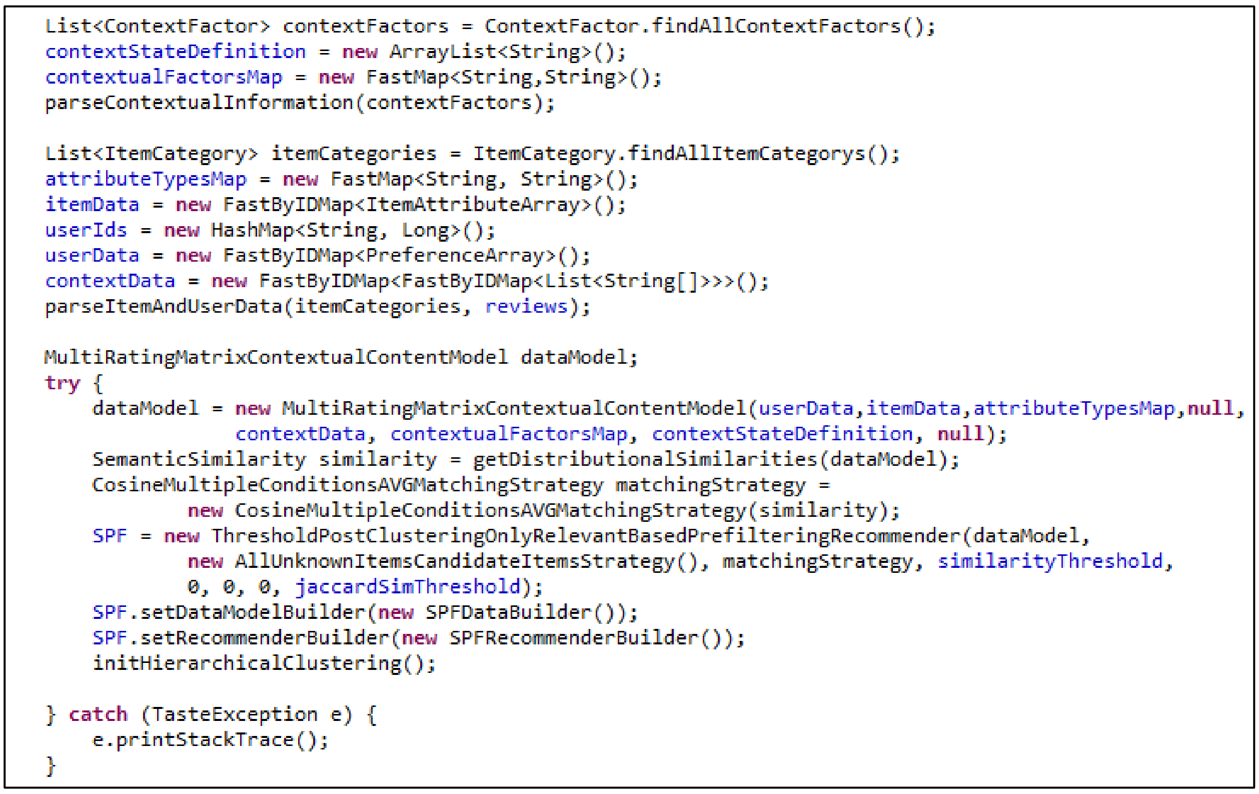
This section illustrates how to use the implemented recommendation algorithms from an existing recommender system, using as an example the integration of the best-performing SPF variant in the *Tourism* data set (according to the evaluation described in Section 5.4) into a mobile tourism recommender system called STS.

In this case the mobile app has been developed in order that the client part has been kept as thin as possible and it works only in a limited way offline. The client comprises a user interface, a presentation logic component as well as a session-handling component. On the server resides the entire recommendation logic which is divided by two asynchronous phases: the learning phase and the recommendation phase. The learning phase is performed offline (every five minutes), and once the model is learned, recommendations can be produced in constant time. Based on these two asynchronous phases, we have implemented a “wrapper” class on the server, logically divided into learning and recommendation phase, which is responsible for transforming the data structures used in STS to the *Apache Mahout*’s data types as well as initializing and using the *Apache Mahout*’s implementation of the SPF variant.

6.4.1. *Learning Phase*

Figure 6.3 shows the code of the learning phase implemented in the wrapper class, where the best-performing SPF variant is initialized. Particularly, the SPF variant that we integrated into STS is the one which combines the following methods:

- Condition’s semantic-vector representation reduced by using the method based on *item categories* (described in Section 5.3.1);
- *Direct* situation-to-situation similarity function (described in Section 5.3.2);
- The relevance-boosted method using the *filtering* strategy (described in Section 0);
- Local model reduction based on *hierarchical clustering* strategy (presented in Section 5.3.4).



```
List<ContextFactor> contextFactors = ContextFactor.findAllContextFactors();
contextStateDefinition = new ArrayList<String>();
contextualFactorsMap = new FastMap<String,String>();
parseContextualInformation(contextFactors);

List<ItemCategory> itemCategories = ItemCategory.findAllItemCategorys();
attributeTypesMap = new FastMap<String, String>();
itemData = new FastByIDMap<ItemAttributeArray>();
userIds = new HashMap<String, Long>();
userData = new FastByIDMap<PreferenceArray>();
contextData = new FastByIDMap<FastByIDMap<List<String[]>>>();
parseItemAndUserData(itemCategories, reviews);

MultiRatingMatrixContextualContentModel dataModel;
try {
    dataModel = new MultiRatingMatrixContextualContentModel(userData,itemData,attributeTypesMap,null,
                contextData, contextualFactorsMap, contextStateDefinition, null);
    SemanticSimilarity similarity = getDistributionalSimilarities(dataModel);
    CosineMultipleConditionsAVGMatchingStrategy matchingStrategy =
            new CosineMultipleConditionsAVGMatchingStrategy(similarity);
    SPF = new ThresholdPostClusteringOnlyRelevantBasedPrefilteringRecommender(dataModel,
            new AllUnknownItemsCandidateItemsStrategy(), matchingStrategy, similarityThreshold,
            0, 0, 0, jaccardSimThreshold);
    SPF.setDataModelBuilder(new SPFDataBuilder());
    SPF.setRecommenderBuilder(new SPFRecommenderBuilder());
    initHierarchicalClustering();

} catch (TasteException e) {
    e.printStackTrace();
}
```

Figure 6.3. Code of the STS wrapper class implementing the SPF learning phase

The *parseContextualInformation(contextFactors)* function transforms the data structure representing the contextual information captured in STS to the data structure used in *Apache Mahout*, which is represented by the objects *contextStateDefinition* (a list of Strings each of which represents a contextual factor) and *contextualFactorsMap* (which stores the conditions-factors mappings using *FastByIDMap*, the faster map implementation used in *Apache Mahout*).

The *parseItemAndUserData (itemCategories, reviews)* function transforms the data structures representing item and contextual user rating data in STS to the data types used in *Apache Mahout*. This function modifies the following objects: *attributeTypesMap*, which is the *FastMap* that stores the item's attributes-values mappings; *itemData*, a *FastByIDMap* which stores the mappings between items and their attributes vector; *userData*, a *FastByIDMap* which stores the mappings between users and their ratings (represented as array of *Preference*); and *contextData*, a *FastByIDMap* which stores the mappings between users, items and contextual situations (note that in this case multiple ratings per user and item are allowed).

Once all the training data are correctly initialized, we create the multi-rating data model (an instance of *MultiRatingMatrixContextualContentModel*). Then, this data model is used to calculate the distributional similarities of all contextual conditions (function *getDistributionalSimilarities*). The resulting *SemanticSimilarity* object is then used to initialize the situation-to-situation similarity function using the *direct* matching strategy. The objects *matchingStrategy* and *dataModel* together with the SPF variant meta-parameters (the global *similarityThreshold* and the *JaccardSimThreshold* that constrains the hierarchical clustering strategy) are then used to create an instance of the SPF recommender.

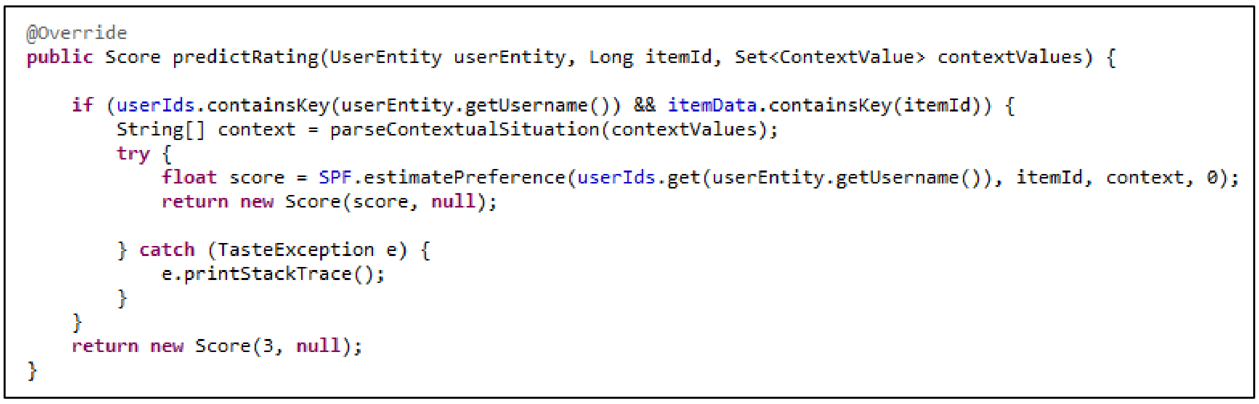
Next, we assign to the SPF recommender the context-free prediction model that will be used to build the target local prediction models. In this case, we use as context-free prediction model *bias MF*, which learns the model parameters from a bi-dimensional *MatrixDataModel*. *SPFDataBuilder* is the constructor responsible for creating each instance of *MatrixDataModel* from a local set of training ratings. *SPFRecommenderBuilder* is the constructors responsible for building an instance of *bias MF* (*BiasBasedMatrixFactorizationRecommender)* for each local *MatrixDataModel*.

Finally, the *initHierarchicalClustering()* function, first creates the semantically expanded local sets of training ratings for each possible contextual situation (in the training set), and then executes the *hierarchical* clustering strategy over this initial sets of ratings, which produces a reduced set of local MF prediction models.

6.4.2. *Recommendation Phase*

After the learning phase is executed, SPF is ready to make recommendations: **Figure 6.4** illustrates the code of the wrapper class implementing the recommendation phase in STS. As it can be observed, STS makes recommendations using SPF if the target user and item exist in the training set; otherwise, a default prediction score equivalent to a 3-star rating is returned.

The function *parseContextualSituation(contextValues)* transforms the data structure in STS representing the target contextual situation (a set of *ContextValue*) to the *Apache Mahout*'s format (an array of *String*). Once correctly formatted the context, SPF makes a prediction for the target user, item and situation. This score is then used by STS system to provide a personalized ranking to the user.



```
@Override
public Score predictRating(UserEntity userEntity, Long itemId, Set<ContextValue> contextValues) {

    if (userIds.containsKey(userEntity.getUsername()) && itemData.containsKey(itemId)) {
        String[] context = parseContextualSituation(contextValues);
        try {
            float score = SPF.estimatePreference(userIds.get(userEntity.getUsername()), itemId, context, 0);
            return new Score(score, null);

        } catch (TasteException e) {
            e.printStackTrace();
        }
    }
    return new Score(3, null);
}
```

Figure 6.4. Code of the STS wrapper class implementing the SPF recommendation phase.

6.5. Summary and Outlook

In this chapter we have described the implementation of the recommendation framework developed in this thesis. We have implemented several recommendation algorithms described in previous chapters on top of *Apache Mahout*, an existing open source recommendation framework, in order to make our experiments more reproducible by other researchers as well as to ease the development of new recommendation algorithms.

We have extended *Apache Mahout* by including several state-of-the-art recommendation algorithms and an additional tool for offline evaluation. We illustrated some usage examples of this tool for evaluating SCB and SPF (the proposed recommendation algorithms in this thesis that we described in Chapters 4 and 5, respectively) and also an example of how to use the developed recommendation framework from an existing recommender system.

The recommendation framework is implemented as an independent distribution of *Apache Mahout* on top of version 0.6. Currently, this extension is available to any practitioner or researcher upon request. However, in the near future, we expect to integrate our extension with the official distribution of *Apache Mahout*.

Chapter 7 - Conclusions

This thesis investigated the effectiveness of exploiting data-driven distributional semantics to address the sparsity-related limitations of state-of-the-art recommendation approaches. Specifically, this research pursued the following two goals:

- To implement and evaluate content-based recommendation algorithms capable of exploiting the distributional semantics of items' attributes to outperform state-of-the-art CB and SERS techniques.
- To implement and evaluate context-aware recommendation algorithms capable of leveraging the distributional semantics of contextual conditions during context modeling to outperform state-of-the-art CARS techniques.

In this thesis, we described and evaluated two novel SERS techniques exploiting distributional semantics: (1) for enhanced CB recommendation, called *Semantic Content-based* (SCB) Filtering, and (2) for enhanced context-aware recommendation, called *Semantic Pre-filtering* (SPF). Finally, we described the implementation of the developed recommendation framework that contains the proposed SERS techniques.

This chapter presents a summary of these contributions (Section 7.1), analyzes their main limitations and presents future research directions to address these limitations (Section 7.2).

7.1. Thesis Contributions

This thesis investigated how distributional semantic similarities between items' attributes and contextual conditions can be exploited in CB and CARS techniques to alleviate the limitations of the state of the art related to the *data-sparsity* problem, and thus improve prediction accuracy.

In distributional semantics, the meaning of a concept is based on its distributional properties, which are automatically derived from the corpus of data where the concept is used. The fundamental idea behind this way to extract semantic similarities between domain concepts is the so-called **distributional hypothesis:** *concepts repeatedly co-occurring in the same context or usage tend to be related.*

In this thesis, we employed distributional-similarity measures to infer semantic similarities between concepts describing entities of the recommendation space. Therefore, concepts are either attributes describing items or conditions describing contextual situations. In the following subsections we summarize the main contributions and present general conclusions of this research.

7.1.1. *Distributional Semantics for Enhanced Content-Based Recommendation*

In this thesis, we have proposed SCB, a SERS approach to CB recommendation that employs a novel strategy to mitigate the lack of semantics of traditional CB approaches. This method consists of exploiting distributional similarities between attributes using a pairwise profile-matching strategy in the prediction phase. Concretely, we proposed two variants of SCB: a *best-pairs* strategy, in which each item's attribute is compared to the most similar user's interest; and an *all-pairs* strategy, where every item's attribute is compared to every user's interest. In both variants we used a global similarity threshold to delimit the minimum attribute-to-attribute similarity to be considered during item-to-user profile matching.

We empirically demonstrated that the *best-pairs* variant is better for rating prediction and the *all-pairs* variant for ranking or top-n recommendation. This difference on performance is explained by the fact that the *all-pairs* strategy tends to estimate "extreme" rating predictions (i.e. predictions closer to 1 or 5 stars) due to the higher number of aggregated pairwise comparisons, which causes bigger mistakes than using predictions more centered to the average (i.e. values close to 3 stars), like the ones of *best-pairs* variant. However, these overestimated scores are less inconvenient for ranking, since in this case prediction accuracy is measured by the position of the recommended items on the ranked list, and not by how close the predicted score is to the true rating.

We also demonstrated that distributional similarities derived from co-occurrences over attribute-based user profiles are more effective for improving the effectiveness of recommendations than similarities derived from item-based co-occurrences or ontology-based measures. This better performance of user-based distributional similarities is due to the fact that the derived semantic associations are based on both users' interests and item's descriptions.

Compared to the state-of-the-art SERSs techniques, we showed that the proposed SCB variants clearly achieve the best results in both recommendation tasks: rating prediction and ranking. Specifically, we compared the proposed pairwise profile-matching strategies to a user-profile expansion strategy based on CSA, as well as to a strategy for latent item-profile representation using LSA. The results also showed that SCB outperforms state-of-the-art MF approaches when recommending to users with few ratings (i.e. cold-start users).

According to the results, the proposed method is a good alternative to ontology-based SERSs, especially when the available domain ontologies consists of general hierarchies which are not expressive enough to infer useful domain-specific relations between items' attributes.

7.1.2. *Distributional Semantics for Enhanced Context-Aware Recommendation*

CARSs assume that, when context matters, the ratings acquired in the target contextual situation are more relevant for predicting what to recommend in that situation. However, these techniques also suffer from the *data-sparsity* problem, since in order to generate accurate recommendation they need a large data set of contextually-tagged ratings, i.e., ratings for items provided in various contextual situations, which are encountered by a user while experiencing an item.

In the thesis, we claim that the *data-sparsity* limitation of state-of-the-art CARS techniques is mainly due to their lack of context-semantics understanding and exploitation during context modeling. We have then proposed a novel SERS approach to context-aware recommendation that implements a solution to the *data-sparsity* problem based on exploiting similarities between contextual situations/conditions in the context modeling phase. Concretely, the proposal (which we named SPF) consists of a reduction-based pre-filtering approach that builds a local MF prediction model for each target contextual situation by reusing ratings tagged with situations semantically related to the target one. We proposed a method to control the level of contextualization (i.e. how many situations are aggregated in the local model) by means of a global similarity threshold, which determines the minimum similarity necessary to consider a contextual situation as reusable in a given target context.

We showed that the effectiveness of SPF depends on the specific method used to compute the situation-to-situation similarity. We evaluated several similarity functions that are all based on the distributional semantics of contextual conditions, i.e., assuming that two situations are similar if they are defined by elementary conditions that influence users' ratings in a similar way. A key component of our approach is the method to estimate the influence of a condition with respect to items or users. We proposed a method especially designed for explicit ratings based on how the true rating deviates from a predicted context-free rating when the contextual condition holds.

Additionally, we improved even more the effectiveness of our approach using a model of the relevance of contextual conditions. This model measures the relevance of a condition as the variance of its corresponding semantic vector representation, tailored to our method to measure the influence of conditions with respect to items or users. We also presented a more efficient and scalable implementation of SPF that reduces the number of local models to build by means of clustering strategies and, at the same time, preserving their predictive accuracy.

According to the results on six different contextually-tagged rating data sets, we showed that SPF, when used in combination with MF local models, clearly outperforms two state-of-the-art CARS based on MF, especially in those data sets where contextual situations have a finer granularity and higher data sparsity.

7.1.3. *Recommendation Framework*

In the existing open-source recommendation frameworks, such as *MyMediaLite*, *Apache Mahout*, *GraphLab* and *LensKit*, there is a lack of support for developing and evaluating CB, CARS and SERS techniques, since the majority only focuses on context-free CF algorithms.

In this thesis, we implemented all the proposed recommendation algorithms as part of the *Apache Mahout* machine-learning library, thus extending it with additional functionality for developing and evaluating CB recommendation algorithms, CARS and SERS techniques. We also included a tool that allows developers to easily perform offline evaluations of all the available recommendation algorithms on data provided in text files. The main motivation for implementing our algorithms and this evaluation tool as part of *Apache Mahout* was to allow other researchers to reproduce the described experiments and make new progress in the RS field easier.

This extended software library provides a state-of-the-art recommendation framework that can be used to easily integrate all the available recommendation algorithms into an existing recommender system. As an example, we described how to use our SPF implementation in a tourism recommender system.

Currently, the recommendation framework is implemented as an independent distribution of *Apache Mahout* on top of version 0.6, and is available to any practitioner or researcher upon request. In the near future, we expect to integrate our extension (or a part of it) with the official *Apache Mahout* distribution.

7.2. Limitations and Future Work

This thesis presented two novel SERS techniques that exploit distributional semantic similarities to alleviate the sparsity-related limitations of current recommendation techniques. The performed experimental evaluations showed that the proposed techniques outperform state-of-the-art approaches in diverse data sets. Here we discuss limitations and further improvements of the proposed techniques as well as promising lines of future research.

Fine-grained meta-parameter optimization. As in other machine learning tasks, it is important to select the optimal meta-parameters of the prediction models in order to maximize their effectiveness in the target application domain. A particularly relevant meta-parameter of the proposed SERS techniques is the *similarity threshold*, which is used to decide which semantic similarities are strong enough to be considered by the model. In this thesis, we proved that using a global similarity threshold optimized for each data set is enough to obtain good results. However, in some data sets, a fine-grained tuning of the similarity threshold could improve even more the accuracy of the models. For example, when using SPF in data sets where ratings are not uniformly distributed among the possible contextual situations, it would be better to find the optimal similarity threshold per each contextual situation or group of situations. Similarly, in SCB a different threshold per user or groups of users with similar profile size could be used. In this way, we would learn optimal thresholds that are adapted to the characteristics of each type of user or context, and not only to the overall data. For instance, for users with few ratings a low threshold could be more adequate in order to increase the contribution of the semantic part, whereas for users with many ratings a high threshold, limiting more the semantic aggregation, could be more adequate.

Besides the similarity thresholds, other meta-parameters can be further improved. Currently, SPF employs the same configuration as the global, context-free MF prediction model (i.e. the same number of latent factors, regularization parameters, etc.) to train the local MF models. In larger data sets, the prediction accuracy could be further improved by learning independent optimal meta-parameters for each local MF model. However, the main difficulty of this fine-grained optimization is how to find the optimum without over-fitting, especially in small-medium rating data sets.

Extending other recommendation techniques with distributional semantics. The proposed methods for acquiring distributional similarities from rating data can be similarly applied in other existing CB and CARS recommendation techniques. For instance, a feature-augmentation or metal-level hybrid recommender could use the distributional similarities between item’s attributes to enhance the item-to-item or user-to-user profile matching. Similarly, a contextual modeling approach extending MF, such as TF [Karatzoglou et al. 2010] and CAMF [Baltrunas et al. 2011b], could exploit situation-to-situation distributional similarities to obtain fine-grained clusters of contextual situations, and thus reducing the *data-sparsity* effects and the number of model parameters to be learnt.

Combining ontology-based and distributional measures. In this thesis, we have shown that distributional similarity measures can be more useful than ontology-based ones to improve the prediction accuracy of recommender systems. However, in very sparse data sets, in which no enough training rating data are available to learn reliable distributional similarities, a combination of ontology-based and distributional measures could improve the precision of the similarity assessments.

Dealing with implicit user data. Although most research on the RS field focuses on making recommendations based on explicit user feedback, the large majority of the data available and easy to collect in real recommender systems is indeed implicit. This type of feedback commonly consists of user’s actions, such as browsing and purchase history. The main limitation of implicit user data, as used today, is that they only provide positive evidence of the user’s interests, and this could affect the accuracy of the distributional similarities used in SPF and SCB. The exploration of SERS techniques using distributional similarities based on implicit user data can be a promising research line for the future.

Appendix A - Acronyms

The following table describes the meaning of the acronyms used throughout this document.

Acronym	Meaning
AI	artificial intelligence
API	application program interface
AUC	area under the ROC curve
BPR	Bayesian personalized ranking
CAMF	context-aware matrix factorization
CARS	context-aware recommender systems
CB	content-based
CF	collaborative filtering
CSA	constrained spreading activation
DCR	differential context relaxation
DCW	differential context weighting
ESA	explicit semantic analysis
IB	item-based
IC	information content
IR	information retrieval
LCS	lowest common subsumer
LDSD	linked data semantic distance
LSA	latent semantic analysis
MAE	mean absolute error
MAP	mean average precision
MD	multidimensional
MF	matrix factorization
ML	machine learning
MRR	mean reciprocal rank
NDCG	normalized discounted cumulative gain
OB	ontology-based
ODP	open directory project
POI	place of interest
PMI	pointwise mutual information
RI	random indexing
RMSE	root mean squared error
RS	recommender systems
SA	spreading activation
SCB	semantic content-based
SERS	semantically-enhanced recommender systems
SGD	stochastic gradient descent

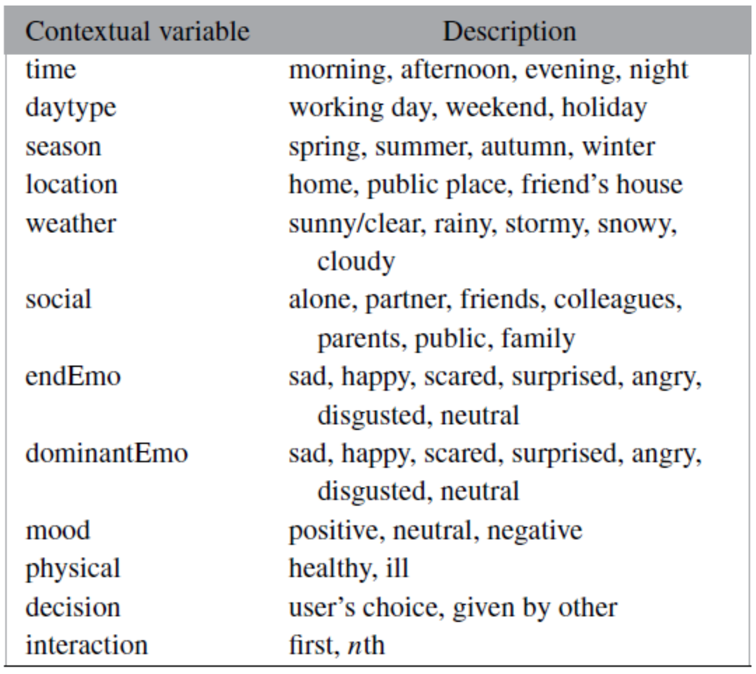
STS	South Tyrol suggests
SPF	semantic pre-filtering
SVD	singular value decomposition
TIPI	ten-item personality inventory
TF	tensor factorization
TF-IDF	term frequency-inverse document frequency
UB	user-based
UI	user interface
VSM	vector space model

Appendix B - Description of Contextual Information

We evaluated SPF on several real-world contextually-tagged data sets. Each of these data sets contains different contextual conditions and factors. Here we present the contextual information that is captured in movie *Comoda* data set (shown in **Table 7.1**) and in *Tourism* data set (shown in **Table 7.2**).

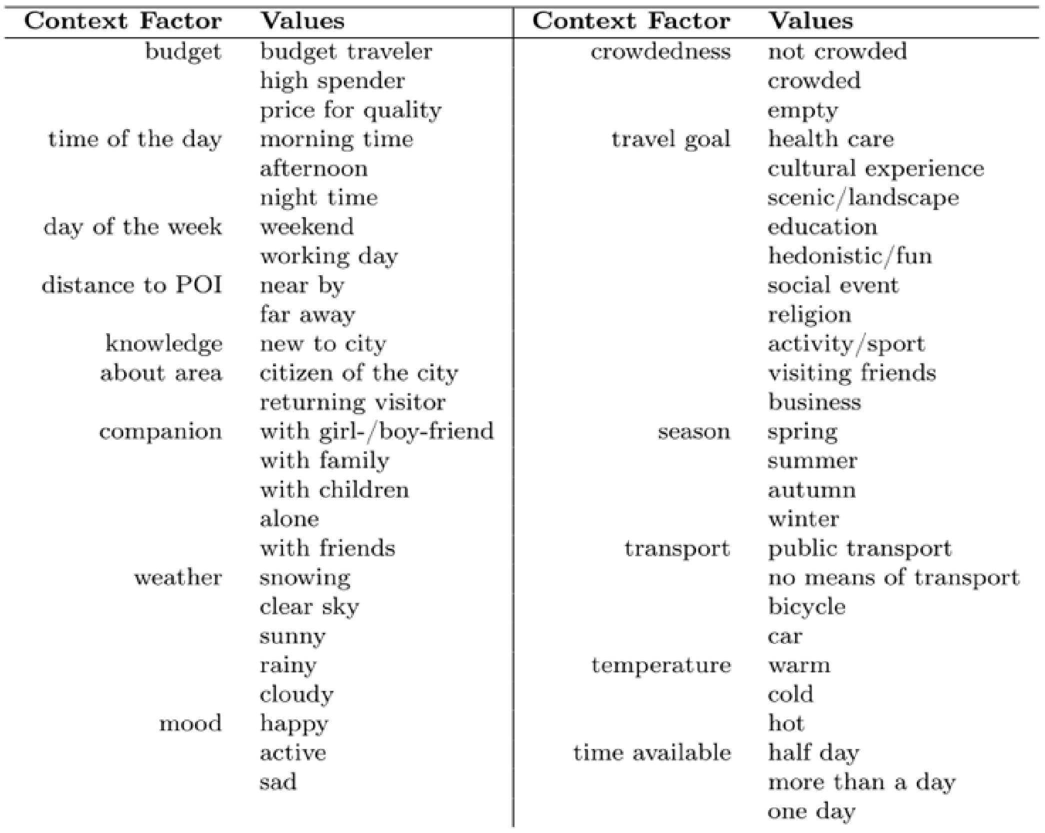
The factors *endEmo* and *dominantEmo* in **Table 7.1** capture the emotional state of a user while watching a movie at different stages of the movie-watching experience: *endEmo* is the emotional state at the end of the movie, and *dominantEmo* is the emotional state that was dominant during the consumption.

Table 7.1. Contextual information represented in *Comoda* data set [Odić et al. 2013]



Contextual variable	Description
time	morning, afternoon, evening, night
daytype	working day, weekend, holiday
season	spring, summer, autumn, winter
location	home, public place, friend's house
weather	sunny/clear, rainy, stormy, snowy, cloudy
social	alone, partner, friends, colleagues, parents, public, family
endEmo	sad, happy, scared, surprised, angry, disgusted, neutral
dominantEmo	sad, happy, scared, surprised, angry, disgusted, neutral
mood	positive, neutral, negative
physical	healthy, ill
decision	user's choice, given by other
interaction	first, *n*th

Table 7.2. Contextual information represented in Tourism data set [Baltrunas et al. 2012]



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

References

Adomavicius, G., Mobasher, B., Ricci, F., & Tuzhilin, A. (2011). Context-aware recommender systems. *AI Magazine*, *32*(3), 67–80.

Adomavicius, G., Sankaranarayanan, R., Sen, S., & Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, *23*(1), 103–145.

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749. doi:10.1109/TKDE.2005.99

Adomavicius, G., & Tuzhilin, A. (2011). Context-aware recommender systems. In *Recommender Systems Handbook* (pp. 217–256).

Ahn, J., Brusilovsky, P., & Grady, J. (2007). Open user profiles for adaptive news systems: help or harm? In *Proceedings of the 16th international conference on World Wide Web* (pp. 11–20). ACM.

Anand, S., & Mobasher, B. (2007). Contextual recommendation. *From Web to Social Web: Discovering and Deploying User and Content Profiles*, *4737*, 142–160.

Ardissono, L., Goy, A., Petrone, G., Segnan, M., & Torasso, P. (2003). Intrigue: Personalized Recommendation of Tourist Attractions for Desktop and Hand Held Devices. *Applied Artificial Intelligence*, *17*(8), 687–714. doi:10.1080/713827254

Balabanovic, M., & Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *In: Communications of the ACM*, *40*(3), 66–72. doi:10.1097/BRS.0b013e3181aa1ffa

Baltrunas, L., & Amatriain, X. (2009). Towards time-dependant recommendation based on implicit feedback. In *Proceedings of the 2009 Workshop on Context-Aware Recommender Systems*.

Baltrunas, L., Kaminskas, M., Ludwig, B., Moling, O., Ricci, F., Aydin, A., … Schwaiger, R. (2011). Incarmusic: Context-aware music recommendations in a car. In *E-Commerce and Web Technologies* (pp. 89–100).

Baltrunas, L., Ludwig, B., Peer, S., & Ricci, F. (2012). Context relevance assessment and exploitation in mobile recommender systems. *Personal and Ubiquitous Computing*, *16*(5), 507–526. doi:10.1007/s00779-011-0417-x

Baltrunas, L., Ludwig, B., & Ricci, F. (2011). Matrix Factorization Techniques for Context Aware. In *RecSys'11* (pp. 301–304).

Baltrunas, L., & Ricci, F. (2009). Context-dependent items generation in collaborative filtering. In *Proceedings of the 2009 Workshop on Context-Aware Recommender Systems*.

Baltrunas, L., & Ricci, F. (2014). Experimental evaluation of context-dependent collaborative filtering using item splitting. *User Modeling and User-Adapted Interaction*, (Special issue on Context-Aware Recommender Systems). doi:10.1007/s11257-012-9137-9

Bambini, R., Cremonesi, P., & Turrin, R. (2011). A recommender system for an iptv service provider: a real large-scale production environment. In *Recommender Systems Handbook* (pp. 299–331).

Beel, J., Langer, S., Genzmehr, M., Gipp, B., & Nürnberger, A. (2013). A Comparative Analysis of Offline and Online Evaluations and Discussion of Research Paper Recommender System Evaluation. In *RepSys'13*.

Billsus, D., & Pazzani, M. J. (1999). A hybrid user model for news story classification. In *Proceeding UM '99 Proceedings of the seventh international conference on User modeling* (pp. 99–108).

Billsus, D., & Pazzani, M. J. (2000). User Modeling for Adaptive News Access. *User Modeling and User-Adapted Interaction*, *10*, 147–180.

Blanco-Fernandez, Y., Pazos-Arias, J., Ramos-Cabrer, M., & Lopez-Nores, M. (2008). Providing entertainment by content-based filtering and semantic reasoning in intelligent recommender systems. *IEEE Transactions on Consumer Electronics*, *54*(2), 727–735. doi:10.1109/TCE.2008.4560154

Bouneffouf, D., Bouzeghoub, A., & Gançarski, A. (2012). A contextual-bandit algorithm for mobile context-aware recommender system. *Neural Information Processing*, *7665*, 324–331.

Braunhofer, M., Elahi, M., Ricci, F., & Schievenin, T. (2014). Context-Aware Points of Interest Suggestion with Dynamic Weather Data Management. In *ENTER 2014*.

Bridge, D., Göker, M. H., MCginty, L., & Smyth, B. (2005). Case-based recommender systems. *The Knowledge Engineering Review*, *20*, 315–320.

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, *12*(4), 331–337.

Burke, R. (2007). Hybrid Web Recommender Systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *Adaptive Web* (pp. 377 – 408).

Campos, P., Díez, F., & Cantador, I. (2014). Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Model User-Adapted Interaction*, (Special issue on Context-Aware Recommender Systems).

Campos, P., Fernández-Tobías, I., Cantador, I., & Díez, F. (2013). Context-Aware Movie Recommendations: An Empirical Comparison of Pre-Filtering, Post-Filtering and Contextual Modeling Approaches. In *EC-WEB*.

Cantador, I., Bellogín, A., & Castells, P. (2008). Ontology-based personalised and context-aware recommendations of news items. In *Proc. of IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology* (pp. 562–565).

Cantador, I., Castells, P., & Bellogín, A. (2011). An Enhanced Semantic Layer for Hybrid Recommender Systems : Application to News Recommendation. *International Journal on Semantic Web and Information Systems (IJSWIS)*, *7*(1), 35.

Chen, A. (2005). Context-aware collaborative filtering system: Predicting the user's preference in the ubiquitous computing environment. *Location-and Context-Awareness*, 244–253.

Chen, L., & Sycara, K. (1998). WebMate: a personal agent for browsing and searching. In *Proceedings of the second international conference on Autonomous agents* (pp. 132–139). ACM.

Codina, V. (2009). *Design, development and deployment of an intelligent, personalized recommendation system. System*. UPC Commons.

Codina, V. (2012). *A Recommender System for the Semantic Web: Application in the tourism domain*. Scholar's Press.

Codina, V., & Ceccaroni, L. (2010a). A Recommendation System for the Semantic Web. In A. Ponce de Leon F. de Carvalho, S. Rodríguez-González, J. F. De Paz, & J. . Corchado (Eds.), *Proceedings of the International Symposium on Distributed Computing and Artificial Intelligence* (pp. 45–52). September 7-10, 2010, Valencia, Spain: Springer.

Codina, V., & Ceccaroni, L. (2010b). Taking advantage of semantics in recommendation systems. In R. Alquézar, A. Moreno, & J. Aguilar (Eds.), *Proceedings of the International Congress of the Catalan Association of Artificial Intelligence* (pp. 163 – 172). October 20-22, 2010, L'Espluga de Francolí, Spain: IOS Press, Amsterdam, The Netherlands.

Codina, V., & Ceccaroni, L. (2011). Extending Recommendation Systems with Semantics and Context-Awareness : Pre-Filtering Algorithms. In C. Fernández, H. Geffner, & F. Manyà (Eds.), *Proceedings of the International Congress of the Catalan Association of Artificial Intelligence* (pp. 81–90). October 26-28, 2011, Lleida, Spain: IOS Press, Amsterdam, The Netherlands.

Codina, V., & Ceccaroni, L. (2012). Semantically-Enhanced Recommenders. In D. Riaño, E. Onaindia, & M. Cazorlam (Eds.), *Proceedings of the International Congress of the Catalan Association of Artificial Intelligence* (pp. 69–78). October 24-26, 2012, Alicante, Spain: IOS Press, Amsterdam, The Netherlands.

Codina, V., Ricci, F., & Ceccaroni, L. (2013a). Exploiting the Semantic Similarity of Contextual Situations for Pre-filtering Recommendation. In S. Carberry, S. Weibelzahl, A. Micarelli, & G. Semeraro (Eds.), *Proceedings of the 21th International Conference on User Modeling, Adaptation, and Personalization (UMAP'13)* (pp. 165–177). June 10-14, Rome, Italy: Springer, Berlin Heidelberg.

Codina, V., Ricci, F., & Ceccaroni, L. (2013b). Local Context Modeling with Semantic Pre-filtering. In *Proceedings of the 7th ACM conference on Recommender systems (RecSys'13)* (pp. 363–366). October 14-16, 2013, Hong Kong: ACM New York, NY, USA.

Codina, V., Ricci, F., & Ceccaroni, L. (2013c). Semantically-enhanced pre-filtering for context-aware recommender systems. In *Proceedings of the 3rd Workshop on Context-awareness in Retrieval and Recommendation* (pp. 15–18). February 5, Rome, Italy: ACM New York, NY, USA.

Codina, V., Ricci, F., & Ceccaroni, L. (submitted). Using Semantic Pre-filtering for Context-Aware Recommendation: an Experimental Evaluation. *User Model User-Adapted Interaction*.

Cremonesi, P., Koren, Y., & Turrin, R. (2010). Performance of recommender algorithms on top-n recommendation tasks. In *RecSys 2010*.

Crestani, F. (1997). Application of spreading activation techniques in information retrieval. *Artificial Intelligence Review*, 453–482.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407.

Degemmis, M., Lops, P., & Semeraro, G. (2007). A content-collaborative recommender that exploits WordNet-based user profiles for neighborhood formation. *User Modeling and User-Adapted Interaction*, *17*(3), 217–255. doi:10.1007/s11257-006-9023-4

Degemmis, M., Lops, P., Semeraro, G., & Basile, P. (2008). Integrating tags in a semantic content-based recommender. *Proceedings of the 2008 ACM Conference on Recommender Systems - RecSys '08*, 163. doi:10.1145/1454008.1454036

Dey, A. (2001). Understanding and using context. *Personal and Ubiquitous Computing*, *5*(1), 4–7.

Dourish, P. (2004). What we talk about when we talk about context. *Personal and Ubiquitous Computing*, *8*(1), 19–30. doi:10.1007/s00779-003-0253-8

Eirinaki, M., Vazirgiannis, M., & Varlamis, I. (2003). SEWeP: using site semantics and a taxonomy to enhance the Web personalization process. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (p. 108). ACM.

Ekstrand, M. D., Ludwig, M., Konstan, J. A., & Riedl, J. T. (2011). Rethinking the Recommender Research Ecosystem: Reproducibility, Openness, and LensKit. In *RecSys'11* (pp. 133–140). New York, NY, USA: ACM. doi:10.1145/2043932.2043958

Ekstrand, M. D., Riedl, J. T., & Konstan, J. A. (2010). Collaborative Filtering Recommender Systems. *Foundations and Trends® in Human–Computer Interaction*, *4*(2), 81–173. doi:10.1561/1100000009

Fink, J., & Kobsa, A. (2002). User Modeling for Personalized City Tours. *Artificial Intelligence Review*, *18*(1), 33–74.

Firth, J. R. (1957). A synopsis of linguistic theory 1930-55. *The Philological Society, 1952-59*, 1–32.

Forbes, P., & Zhu, M. (2011). Content-boosted matrix factorization for recommender systems: experiments with recipe recommendation. In *RecSys'11* (pp. 23–26).

Funk, S. (2006). Netflix update: Try this at home. *http://sifter.org/˜simon/journal/20061211.html*.

Gabrilovich, E., & Markovitch, S. (2007). Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (pp. 6–12).

Ganesan, P., Garcia-Molina, H., & Widom, J. (2003). Exploiting hierarchical domain structure to compute similarity. *ACM Transactions on Information Systems (TOIS)*, *21*(1), 64–93.

Gantner, Z., Rendle, S., Freudenthaler, C., & Schmidt-thieme, L. (2011). MyMediaLite : A Free Recommender System Library. In *RecSys'11*.

Gunawardana, A., & Meek, C. (2008). Tied Boltzmann Machines for Cold Start Recommendations. In *Recsys'08*.

Hariri, N., Mobasher, B., & Burke, R. (2012). Context-aware Music Recommendation Based on Latenttopic Sequential Patterns. In *Proceedings of the Sixth ACM Conference on Recommender Systems* (pp. 131–138). New York, NY, USA: ACM. doi:10.1145/2365952.2365979

Hayes, C., & Cunningham, P. (2004). Context boosting collaborative recommendations. *Knowledge-Based Systems*, *17*(2-4), 131–138. doi:10.1016/j.knosys.2004.03.008

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

Hidasi, B., & Tikk, D. (2012). Fast ALS-Based Tensor Factorization for Context-Aware Recommendation. In *KDD'12* (pp. 67–82).

Hill, W., Stead, L., Rosenstein, M., & Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 194–201).

Järvelin, K., & Kekäläinen, J. (2002). Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inf. Syst.*, *20*(4), 422–446. doi:10.1145/582415.582418

Jiang, J., & Conrath, D. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. In *Research and Computational Linguistics*.

Johnson, W., & Lindenstrauss, J. (1984). Extensions of {L}ipschitz mappings into a {H}ilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)* (Vol. 26, pp. 189–206). American Mathematical Society.

Kaminskas, M., Fernández-Tobías, I., Ricci, F., & Cantador, I. (2012). Knowledge-based music retrieval for places of interest. *Proceedings of the Second International ACM Workshop on Music Information Retrieval with User-Centered and Multimodal Strategies - MIRUM '12*, 19–24. doi:10.1145/2390848.2390854

Kanerva, P., Kristofersson, J., & Holst, A. (2000). Random indexing of text samples for latent semantic analysis.

Karatzoglou, A., Amatriain, X., Baltrunas, L., & Oliver, N. (2010). Multiverse Recommendation: N-dimensional Tensor Factorization for Context-aware Collaborative Filtering. In *RecSys'10* (pp. 79–86).

Karypis, G. (2001). Evaluation of Item-Based Top- N Recommendation. In *Evaluation of item-based top-N recommendation algorithms* (pp. 247–254).

Katz, G., Shani, G., Shapira, B., & Rokach, L. (2012). Using Wikipedia to Boost SVD Recommender Systems. *Computing Research Repository (CoRR).*

Kim, S., Han, K., Rim, H., & Myaeng, S. H. (2006). Some Effective Techniques for Naive Bayes Text Classification. *IEEE Transactions on Knowledge and Data Engineering*, *18*(11), 1457–1466. doi:10.1109/TKDE.2006.180

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). GroupLens: applying collaborative filtering to Usenet news. *Communications of the ACM*, *40*(3), 87.

Koren, Y. (2008). Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 426–434). New York, NY, USA: ACM. doi:10.1145/1401890.1401944

Koren, Y. (2009). The bellkor solution to the netflix grand prize. *Netflix Prize Documentation*, (August), 1–10.

Koren, Y. (2010). Collaborative filtering with temporal dynamics. *Communications of the ACM*, *53*(4), 89. doi:10.1145/1721654.1721677

Koren, Y., & Bell, R. (2011). Advances in collaborative filtering. *Recommender Systems Handbook*, 145–186.

Kullback, S., & Leibler, A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*.

Kurucz, M., Benczúr, A., & Csalogány, K. (2007). Methods for large scale SVD with missing values. *Proceedings of KDD Cup and Workshop*, 31–38.

Leacock, C., & Chodorow, M. (1998). Combining local context and WordNet similarity for word sense identification. In *WordNet: An Electronic Lexical Database* (pp. 265–283).

Lees-Miller, J., Anderson, F., Hoehn, B., & Greiner, R. (2008). Does Wikipedia Information Help Netflix Predictions? In *Seventh International Conference on Machine Learning and Applications* (pp. 337–343). Ieee. doi:10.1109/ICMLA.2008.121

Li, Y., Bandar, Z. a., & McLean, D. (2003). An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, *15*(4), 871–882. doi:10.1109/TKDE.2003.1209005

Lin, D. (1998). An information-theoretic definition of similarity. In *Proceedings of the 15th International Conference on Machine Learning* (Vol. 1, pp. 296–304). Citeseer.

Linden, G., Smith, B., & York, J. (2003). Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, *7*(1).

Liu, H., & Maes, P. (2005). Interestmap: Harvesting social network profiles for recommendations. In *Proceedings of the Workshop on Beyond Personalization* (pp. 54–60).

Liu, L., Lecue, F., Mehandjiev, N., & Xu, L. (2010). Using Context Similarity for Service Recommendation. *2010 IEEE Fourth International Conference on Semantic Computing*, 277–284. doi:10.1109/ICSC.2010.39

Liu, P., Nie, G., & Chen, D. (2007). Exploiting semantic descriptions of products and user profiles for recommender systems. In *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on* (pp. 179–185). IEEE.

Lombardi, S., Anand, S., & Gorgoglione, M. (2009). Context and Customer Behavior in Recommendation. In *RecSys09 Workshop on Context-aware Recommender Systems*.

Lops, P., Gemmis, M. De, & Semeraro, G. (2011). Content-based Recommender Systems : State of the Art and Trends. In *Recommender Systems Handbook* (pp. 73–105). doi:10.1007/978-0-387-85820-3

Low, Y., Gonzalez, J., & Kyrola, A. (2010). Graphlab: A new framework for parallel machine learning. In *26th Conference on Uncertainty in Artificial Intelligence (UAI)*.

Magnini, B., & Strapparava, C. (2001). Improving User Modelling with Content-Based Techniques. *Word Journal Of The International Linguistic Association*, 74–83.

Manning, C., Raghavan, P., & Schütze, H. (2009). *An Introduction to Information Retrieval*. Cambridge Univ Press.

McCallum, A., & Nigam, K. (1998). A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*.

Melville, P., Mooney, R. J., & Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 187–192). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Middleton, S. E., Shadbolt, N. R., & Roure, D. C. D. E. (2004). Ontological User Profiling in Recommender Systems. *ACM Transactions on Information Systems*, *22*(1), 54–88.

Miller, G. A. (1995). WordNet : A Lexical Database for English. *Communications of the ACM*, *38*(11), 39–41.

Mobasher, B., Jin, X., & Zhou, Y. (2003). Semantically Enhanced Collaborative Filtering on the Web. In *Web Mining: from Web to Semantic Web (EWMF 2003)* (pp. 57–76).

Mohammad, S., & Hirst, G. (2012). Distributional measures as proxies for semantic relatedness. *Cornell University Library, 1203.1809*.

Musto, C., Narducci, F., & Semeraro, G. (n.d.). Myusic: a Content-based Music Recommender System based on eVSM and Social Media. In *IIR 2013*.

Nelder, J., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal*, *7*(4), 308–313.

Nguyen, T., & Riedl, J. (2013). Predicting Users' Preference from Tag Relevance. In *User Modeling, Adaptation, and Personalization* (pp. 274–280).

Odić, A., Tkalčič, M., Tasic, J., & Košir, A. (2013). Predicting and detecting the relevant contextual information in a movie-recommender system. *Interacting with Computers*, 1–17.

Ostuni, V., Noia, T. Di, Mirizzi, R., Romito, D., & Sciascio, E. Di. (2012). Cinemappy: a Context-aware Mobile App for Movie Recommendations boosted by DBpedia. In *SeRSy'12*.

Owen, S., Anil, R., Dunning, T., & Friedman, E. (2011). *Mahout in action*.

Panniello, U., & Gorgoglione, M. (2010). Does the recommendation task affect a CARS performance? In *Proceedings of the 2010 Workshop on Context-Aware Recommender Systems*.

Panniello, U., Tuzhilin, A., & Gorgoglione, M. (2014). Comparing context-aware recommender systems in terms of accuracy and diversity: which contextual modeling, pre-filtering and post-filtering methods perform the. *User Model User-Adapted Interaction*, (Special issue on Context-Aware Recommender Systems).

Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C., & 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* (pp. 265–268). ACM.

Passant, A. (2010). Measuring Semantic Distance on Linking Data and Using it for Resources Recommendations. In *AAAI Spring Symposium: Linked Data Meets Artificial …* (pp. 93–98).

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

Pazzani, M., & Billsus, D. (2007). Content-based recommendation systems. In *The Adaptive Web: Methods and Strategies of Web Personalization* (pp. 325–341). Retrieved from http://link.springer.com/chapter/10.1007/978-3-540-72079-9_10

Pazzani, M. J., Muramatsu, J., Billsus, D., & others. (1996). Syskill & Webert: Identifying interesting web sites. In *Proceedings of the national conference on artificial intelligence* (pp. 54–61). AAAI Press / MIT Press.

Rajaraman, A., & Ullman, J. (2012). Clustering. In *Mining of massive datasets* (pp. 239–278).

Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-thieme, L. (2009). BPR : Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 452–461).

Rendle, S., Gantner, Z., Freudenthaler, C., & Schmidt-Thieme, L. (2011). Fast context-aware recommendations with factorization machines. In *SIGIR '11*. New York, New York, USA: ACM Press. doi:10.1145/2009916.2010002

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). GroupLens: An Open Architecture For Collaborative Filtering Of Netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW '94* (pp. 175–186). Chapel Hill, North Carolina: ACM Press. doi:10.1145/192844.192905

Resnik, P. (1995). Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Ricci, F., & Nguyen, Q. N. (2006). MobyRek: A Conversational Recommender System for On-the-move Travellers. In *Destination Recommendation Systems: Behavioural Foundations and Applications* (pp. 281–294).

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). *Recommender Systems Handbook*. (F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor, Eds.). Boston, MA: Springer US. doi:10.1007/978-0-387-85820-3

Rocchio, J. (1971). Relevance Feedback Information Retrieval. In G. Salton (Ed.), *The SMART Retrieval System -Experiments in Automatic Document Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Rubenstein, H., & Goodenough, J. B. (1965). Contextual Correlates of Synonymy. *Commun. ACM*, *8*(10), 627–633. doi:10.1145/365628.365657

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Application of dimensionality reduction in recommender system-a case study. In *In ACM WebKDD 2000 Workshop*.

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

Schütze, H., & Pedersen, J. (1995). Information Retrieval based on Word Senses. In *Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval* (pp. 161–175). Las Vegas, USA.

Sen, S., Vig, J., & Riedl, J. (2009). Tagommenders: connecting users to items through tags. In *Proceedings of the 18th international conference on World wide web* (pp. 671–680). ACM.

Shani, G., & Gunawardana, A. (2011). Evaluating Recommendation Systems. In *Recommender Systems Handbook* (pp. 257–297).

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

Shi, Y., Karatzoglou, A., Baltrunas, L., & Larson, M. (2012). TFMAP : Optimizing MAP for Top-N Context-aware Recommendation. In *SIGIR'12* (pp. 155–164).

Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., & Hanjalic, A. (2012). CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-more Filtering. In *Proceedings of the Sixth ACM Conference on Recommender Systems* (pp. 139–146). New York, NY, USA: ACM. doi:10.1145/2365952.2365981

Shi, Y., Larson, M., & Hanjalic, A. (2013). Mining contextual movie similarity with matrix factorization for context-aware recommendation. *ACM Transactions on Intelligent Systems and Technology*, *4*(1), 1–19. doi:10.1145/2414425.2414441

Shoval, P., Maidel, V., & Shapira, B. (2008). An Ontology Content-Based filtering method. *Information Theories & Applications*, *15*, 303–314.

Sieg, A., Mobasher, B., & Burke, R. (2007). Learning Ontology-Based User Profiles: A Semantic Approach to Personalized Web Search. *IEEE Intelligent Informatics Bulletin*, *8*(1).

Sieg, A., Mobasher, B., & Burke, R. (2010). Improving the Effectiveness of Collaborative Recommendation with Ontology-Based User Profiles. In *RecSys'10*.

Smyth, B. (2007). Case-Based Recommendation. In *The Adaptive Web* (Vol. 4321, pp. 342–376). Berlin: Springer Berlin Heidelberg. doi:10.1007/978-3-540-72079-9_11

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, *25*(1–2), 161–197. doi:http://dx.doi.org/10.1016/S0169-023X(97)00056-6

Van Setten, M., Pokraev, S., & Koolwaaij, J. (2004). Context-aware recommendations in the mobile tourist application COMPASS. In *Adaptive Hypermedia and Adaptive Web-Based Systems* (pp. 515–548). Springer.

Vargas, S., Castells, P., & Vallet, D. (2011). Intent-Oriented Diversity in Recommender Systems. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (pp. 1211–1212).

Wang, Y., Wang, S., Stash, N., Aroyo, L., & Schreiber, G. (2010). Enhancing Content-based Recommendation with the Task Model of Classification. *Lecture Notes in Computer Science*, *6317*, 431–440. doi:0.1007/978-3-642-16438-5_33

Weimer, M., Karatzoglou, A., Le, Q., & Smola, A. (2007). Cofi rank-maximum margin matrix factorization for collaborative ranking. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems* (pp. 1–8). Vancouver, British Columbia, Canada.

Witten, I. H., & Bell, T. C. (1991). The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *Information Theory, IEEE Transactions on*, *37*(4), 1085–1094.

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3rd ed.). Amsterdam: Morgan Kaufmann.

Wu, Z., & Palmer, M. (1994). Verbs Semantics and Lexical Selection. In *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics* (pp. 133–138). Stroudsburg, PA, USA: Association for Computational Linguistics. doi:10.3115/981732.981751

Zhang, M., & Hurley, N. (2008). Avoiding monotony: improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM conference on Recommender systems* (pp. 123–130). ACM.

Zheng, Y., Burke, R., & Mobasher, B. (2012). Optimal feature selection for context-aware recommendation using differential relaxation. In *RecSys 2012 Workshop on Context-Aware Recommender Systems*.

Zheng, Y., Burke, R., & Mobasher, B. (2013a). Recommendation with differential context weighting. In *UMAP 2013* (pp. 152–164).

Zheng, Y., Burke, R., & Mobasher, B. (2013b). The Role of Emotions in Context-aware Recommendation. In *RecSys 2013 Workshop on Human Decision Making in Recommender Systems* (pp. 21–28).