Beyond Accuracy in Recommender Systems under the Linked Data lens

Politecnico di Bari

Department of Electrical and Information Engineering
ELECTRICAL AND INFORMATION ENGINEERING
Ph.D. Program
SSD: ING-INF/05
Final Dissertation

Beyond Accuracy in Recommender Systems under the Linked Data lens

---

by
Paolo Tomeo

Referees:
Prof. Iván Cantador
Prof. Pasquale Lops

Supervisor:
Prof. Tommaso Di Noia

Co-ordinator of Ph.D. Program:
Prof. Vittorio Passaro

XXIX cycle, 2014-2016

To my family

Contents

Abstract v

1 Introduction 1

1.1 Motivation . . . 1
1.2 Research Questions . . . 4
1.3 Contributions . . . 5
1.4 Overview . . . 7
1.5 List of publications . . . 9

2 Recommender Systems 12

2.1 Overview of Recommender Systems . . . 12
2.2 Collaborative Filtering Recommendation . . . 14
2.3 Content-based Recommendation . . . 20
2.4 Hybrid Recommendation . . . 21
2.5 Collaborative Filtering with Side Information . . . 22
2.6 Graph-based Recommender Systems . . . 24
2.6.1 Heterogeneous Information Network . . . 24
2.6.2 HeteRec . . . 26
2.6.3 PathRank . . . 27
2.7 Linked Data for Recommender Systems . . . 28
2.7.1 Feeding Recommender Systems with LD . . . 28
2.7.2 SPrank . . . 29
2.8 Recommender Systems Evaluation . . . 33
2.8.1 Accuracy Metrics . . . 34

2.9 Diversity in Recommender Systems . . . 35
2.9.1 Greedy Selection Algorithms . . . 39
2.9.2 Maximal Marginal Relevance . . . 41
2.9.3 Explicit Query Aspect Diversification . . . 41
2.9.4 Diversity Evaluation Metrics . . . 42
2.10 Cold Start Problem . . . 45

3 Adaptive Multi-attribute Diversity 48
3.1 Introduction . . . 49
3.2 Related work . . . 52
3.3 Adaptive multi-attribute diversification . . . 53
3.3.1 User Quadrants . . . 56
3.3.2 Fuzzy Quadrants . . . 58
3.3.3 Adaptive MMR . . . 59
3.3.4 Adaptive xQuAD . . . 61
3.4 Experimental setting . . . 61
3.4.1 Datasets . . . 61
3.4.2 Recommendation Algorithms . . . 63
3.4.3 Preliminary insight into Movies Recommendation . . . 64
3.5 Experimental Results . . . 65
3.5.1 Comparative Results for MMR . . . 67
3.5.2 Comparative Results for xQuAD . . . 70
3.5.3 Results discussion . . . 71
3.6 Summary . . . 73

4 Regression Trees for Diversity 80
4.1 Introduction . . . 80
4.2 Related work . . . 82
4.3 Intent-aware Multi-attribute Diversity . . . 83
4.4 Experimental setting . . . 86
4.5 Results Discussion . . . 88
4.6 Summary . . . 92

5 Diversification with Temporal Dynamics 94
5.1 Introduction . . . 94
5.2 Related Work . . . 96
5.3 Intent modeling with Temporal Dynamics . . . 96
5.3.1 Time-Based Intent Modeling . . . 97
5.3.2 Session-Based Intent Modeling . . . 98
5.4 Experimental setting . . . 100
5.5 Results Discussion . . . 102
5.6 Summary . . . 103

6 LD and Cross-Domain For Cold-Start 107
6.1 Introduction . . . 108
6.2 Related Work . . . 109
6.3 Dataset . . . 110
6.3.1 Linking Items to DBpedia Entities . . . 111
6.3.2 Final Semantically Annotated Dataset . . . 113
6.3.3 Semantically Enriched Item Profiles . . . 114
6.4 Experimental setting . . . 114
6.5 Results . . . 116
6.6 Summary . . . 123

7 Graph-based Similarity Metrics for CBRS 124
7.1 Introduction . . . 125
7.2 Related Work . . . 126
7.3 Graph-based Semantic Similarity Metrics . . . 128
7.4 Experimental setting . . . 131
7.5 Results Discussion . . . 134
7.6 Summary . . . 136

8 Comparative Analysis of DBpedia and Freebase 138
8.1 Introduction . . . 139
8.2 Related Work . . . 140

8.3 Feature-based Semantic Similarity Measurement 142
8.4 Experimental Setting 145
8.5 Results Discussion 147
8.6 Summary 157

9 Conclusions **159**
9.1 Introduction 159
9.2 Summary and Contributions 160
9.2.1 Adaptive Multi-Attribute Diversity 160
9.2.2 Regression Trees for Multi-Attribute Diversity 161
9.2.3 Diversification with Temporal Dynamics 161
9.2.4 LOD and Cross-Domain For Cold-Start 162
9.2.5 Graph-based Similarity Metrics 162
9.2.6 Comparative Analysis of DBpedia and Freebase 163
9.3 Future Work 163

Bibliography **165**

Appendices **189**

A Semantic Web technologies **190**
A.0.1 Resource Description Framework - RDF 190
A.0.2 Simple Protocol and RDF Query Language - SPARQL 193

B Further results - Chapter 4 **196**

C Further results - Chapter 5 **205**

Abstract

Recommender Systems have become fundamental tools in helping users to find what is relevant for them in situations where information overload makes such task hard or even impossible. Recommender Systems are designed to suggest unknown items to the users in a personalized way, recommending those items that are most likely of interest to the users. While new algorithms and approaches have been proposed over the years mainly devoted to maximizing recommendation accuracy, recently it has been recognized that the predictive accuracy is not enough to guarantee satisfying user experience. Attention has been paid to other important quality factors such as diversity and novelty of the recommendations, and to further issues in this area, for instance the user cold start problem.

At the same time, the Web has evolved from a global information space of linked documents to a Web of Data. The Linked Data initiative born in order to provide a standardized set of best practices for publishing and connecting structured data on the Web, has played a fundamental role in the development of the Web of Data. Semantic data in the Linked Data sources enable the design of new generation of knowledge-driven applications and services.

This thesis investigates a set of research lines in the field of Recommender Systems using Linked Data with a focus on different quality dimensions of recommendations, besides accuracy. Specifically, we propose new methods for personalizing the diversification of list of recommendations over different item dimensions, and a new method for exploiting temporal information in intent-aware diversification. Moreover, we investigate the use of semantic data and cross-domain information for tackling the user cold-start problem. Finally, we compare different semantic similarity metrics and Linked Data

sources to assess their performance in feeding content-based recommender systems.

Experimental results, showed and discussed in this thesis, support the validity of our contributions and analyses.

Chapter 1

Introduction

1.1 Motivation

This period, known as **Information Overload** era, is characterized by a rapid growth of massive amount of information available on Internet, that exceeds the users capability of processing and using it [149]. This problem is also called Infobesity or Infoxication, indicating the negative impact on individual life quality of the current and diffused consuming of large amounts of information, mostly of little interest for the user or redundant. Figure 1.1 shows some impressive numbers about the content published online every minute: about 3.1 million searches on Google, more than 422,000 tweets on Twitter, about 400 hours of video uploaded on YouTube, to cite a few.

Although users might potentially find online anything they may be looking for, in practice they cannot access and profitably use such massive information without the support of intelligent systems, such as automatic filtering tools. Such phenomenon is know as Paradox Choice [141]: huge and fast growing number of possibilities overwhelms the users, leading them to make

Figure 1.1: Statistics of the global social media activity every minute. Chart realized by Smart Insights (www.smartinsights.com/internet-marketing-statistics/happens-online-60-seconds).

poor decisions and feel anxiety and unsatisfaction.

Recommender Systems (RSs) [127] are a family of information filtering tools which have proven to be valuable means in assisting users to find, in a personalized manner, what is relevant for them in such overflowing complex information spaces. They provide users with personalized access to large collections of resources. As a result, such systems have been proposed as essential tools in assisting users to face the information overload problem and applied across several domains [21], such as music [87], TV programs

[15], taxi suggestion [70], digital libraries [179], just to cite a few of them.

While those systems have acquired popularity in both academia and industry, research in this field has highlighted the importance of exploiting knowledge for providing the users with better recommendations [108, 109, 112, 115, 104]. In the same period the Web has evolved from a global information space of linked documents to a **Web of Data** where both documents and data are linked, characterized by self-describing relations *understandable* from machines [20]. The **Linked Data** project born in order to provide a standardized set of best practices for publishing and connecting structured data on the Web [64] is revolutionising access, discovery, integration and use of data, enabling the design of new generation of knowledge-driven applications and services [115, 108, 113].

Since the main task of a recommendation engine is to suggest unknown items in a personalized way and recommend the top N items by considering the highest predicted ratings, in the recommender systems field new algorithms and approaches have been proposed over the years mostly devoted to maximizing recommendation accuracy [23]. However, it has been recognized that predictive accuracy of recommendations is not enough to judge the effectiveness of a recommender system and the resulting user experience [29, 156, 46, 2, 6, 32]. The most accurate recommendations for a user are often too similar to each other and attention has to be paid towards the goal of improving **diversity** in recommended items thus avoiding monotony. More recently, a user study pointed out the strong correlation between perceived accuracy and satisfaction of the users for algorithms able to better diversify the returned list of recommended items [46].

Furthermore, even though recommender systems are useful tools to address the information overload problem, providing large sets of recommendations can still make the users experience a wearing cognitive effort for choosing the appropriate suggestions. This problem is known as *choice overload* in recommender systems and can be faced providing the user with less and preferably diverse recommendations [22].

1.2 Research Questions

The general aim of this thesis is to assess the exploitation of knowledge extracted from Linked Data sources for the recommendation task considering important quality dimensions, such as recommendation accuracy and diversity. In particular, we have pursued the following research goals.

- **RG1 proposal of new methods for multi-attribute diversification.** Multi-attribute content search and filtering techniques have been proposed to help users in better specifying or eliciting her preferences or needs across different attributes. However, the problem of considering different attributes in the diversification of recommendations is mostly under-explored.

- **RG2 proposal of new methods for personalization in multi-attribute diversification.** Experimental evidence shows that different users can require different degrees of diversity across different content-based attributes. New methods for identifying such needs and personalizing the diversification can improve the user experience.

- **RG3 proposal of new methods for exploiting temporal dynamics in intent-aware diversification.** The importance of analyzing temporal dynamics for user modeling has been strongly demonstrated. Therefore, temporal analysis of user activities in intent-aware diversification may lead to better trade-off between accuracy and diversity of the recommendations.

- **RG4 exploration of the use of Linked Data for the tackling the user cold-start problem.** Finding accurate recommendations for users in cold-start situations is one of the most important challenge in the recommender systems field. Generally, additional data is used to compensate the scarcity of user feedback. Knowledge extracted from Linked Data sources can be used to compensate the lack of historical information.

- **RG5 exploration of the use of Linked Data for the implementation of content-based recommender systems.** The quality of content-based recommender systems strongly relies on the quantity and quality of available content information. Linked Data sources can be useful to implement this kind of systems in case of lack of content information. Moreover, the use of Linked Data sources provide different advantages, such as the existence of a consistent and easily accessible amount of knowledge in different domains.

1.3 Contributions

The work we present in this thesis has resulted in several contributions to the research in Recommender Systems field mainly focused on the diversity problem and the use of metadata extracted from Linked Data sources. Main contributions of the work are:

- **Exploitation of Linked Data to describe the items for content-based diversification and diversity evaluation.** For diversifying or to evaluate the degree of diversity of a set of recommendations, information associated to the items is needed. In particular in this thesis we focused on the use of information extracted from Linked Data sources, such as DBpedia and Freebase. It represents a valuable alternative when items' descriptive content is not present. In particular the works presented in Chapters 3 and 4 use LD for both diversification and individual diversity evaluation, while the work in Chapters 5 and 6 use LD only for the evaluation.

- **Presentation of novel approaches for considering multiple attributes in the diversification process.** Since multi-attribute diversity has been substantially non-treated in the literature of recommender systems, we face the problem in Chapter 3, where we propose an adaptation of two well-known diversification algorithms (`MMR` and

xQuAD). While in Chapter 4 we propose a novel intent-aware diversification method based on regression trees that can be used to improve personalized diversification of the recommendation list in a multi-attribute setting.

- **Presentation of a new method for personalizing the recommendation diversity considering the individual inclination of the user to diversifying over different item dimensions.** The proposal includes a modeling method of user propensity towards diversified recommendations and an adaptation of two diversification algorithms (MMR and xQuAD) able to include such user models to personalize the recommendation diversity. In particular such proposal is described in Chapter 3.

- **Presentation of novel user intent modeling methods for intent-aware diversification framework.** Intent-aware diversification aims at improve the diversity degree of a set of recommendations taking into account the user interests. xQuAD is one of the most known intent-aware algorithm. In particular, Chapter 4 presents a method that exploits regressions trees for modeling the user intents over different items attributes, while Chapter 5 introduces a intent modeling method based on temporal dynamics.

- **Exploitation of Linked Data and cross-domain information for tackling the user cold-star problem.** In case of new users, scarcity of feedback makes the recommendation task even more challenging. Additional data can be used to compensate such scarcity. For instance, cross-domain information is useful to extend the user profile, while content-based information allows the use of hybrid RSs. In particular, in Chapter 6 we present a detailed comparison of several recommendations algorithms in terms of accuracy and diversity, including some graph-based algorithms able to combine cross-domain information and knowledge graphs extracted from Linked Data sources.

- **Exploitation of Linked Data for implementing Content-based Recommender Systems.** In Chapter 7 we present a comparative experimental evaluation between the use of the two graph-based similarity metrics SimRank and PageRank for feeding a content-based recommender system, while in Chapter 8 a comparative experimental evaluation between the use of the two Linked Data datasets DBpedia and Freebase as knowledge bases for semantic-aware content-based recommender systems.

- **Evaluation of different quality dimensions of recommendations.** All the experimental evaluations carried out in the work of this thesis include different accuracy and diversity metrics.

1.4 Overview

This thesis is organized as follows:

- **Chapter 1** presents motivation, contributions and publications related to the thesis.

- **Chapter 2** presents the state of the art in recommender systems, describing the main classes of recommendation approaches: collaborative filtering, content based, and hybrid approaches. Then, it focuses on the description of different hybrid approaches, in particular collaborative filtering with side information, graph-based approaches able build upon heterogeneous information network, and finally of the use of linked data for feeding recommender systems. Finally, it presents the state of the art about the evaluation of the recommendation quality and then the diversity problem in recommender systems, the most known diversification algorithms, and the diversity evaluation metrics.

- **Chapter 3** describes a new adaptive multi-attribute diversification method proposed, built upon two different state-of-the-art diversification methods - MMR and xQuAD. It also presents the experimental

evaluation in the movie and book domains, in terms of Accuracy, Individual Diversity, Aggregate Diversity, and Novelty.

- **Chapter 4** describes a novel user modeling technique based on regression trees for intent-aware diversification in multi-attribute content settings. Finally the results of the experimental evaluation show the performance of such approach with respect to both accuracy and diversity measures.

- **Chapter 5** presents two novel intent modeling methods based on temporal dynamics, that can be used for proving better intent-aware diversification of recommendations. One of them is based on a temporal decay function, while the other one exploits a new session-based intent modeling.

- **Chapter 6** presents a comparison of several recommendation methods in using cold-start situations in single and cross-domain scenarios, including two graph-based methods able to exploit heterogeneous information.

- **Chapter 7** presents a comparison of two existing metrics - SimRank and PageRank - and investigate their suitability and performance for computing similarity between resources in Linked Data graphs and investigate their usage to feed a content-based recommender system.

- **Chapter 8** presents a comparison of two well-known Linked Data sources - DBpedia and Freebase - as knowledge graphs to feed content-based recommender systems. In particular, we tested four different recommendation approaches exploiting both DBpedia and Freebase in the music domain.

- **Chapter 9** describes all the conclusions of the work of this thesis and the potential future work.

1.5 List of publications

The work carried out and the results obtained for the realization of this thesis are part of various publications which are listed below.

Journals

- Tommaso Di Noia, Jessica Rosati, Paolo Tomeo, Eugenio Di Sciascio. Adaptive Multi-attribute Diversity for Recommender Systems. Elsevier Information Sciences 2016.
- Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, Eugenio Di Sciascio. SPRank: Semantic Path-based Ranking for Top-N Recommendations using Linked Open Data. ACM Transactions on Intelligent Systems and Technology, 2016.
- Tommaso Di Noia, Vito Claudio Ostuni, Jessica Rosati, Paolo Tomeo, Eugenio Di Sciascio, Roberto Mirizzi, Claudio Bartolini. Building a relatedness graph from Linked Open Data: a case study in the IT domain. Expert Systems with Applications, 2016.

International Conferences

- Aleksandra Karpus, Tommaso Di Noia, Paolo Tomeo, Krzysztof Goczyla. Rating Prediction with Contextual Conditional Preferences. 8th International Conference on Knowledge Discovery and Information Retrieval (KDIR 2016). Porto, Portugal, November, 2016.
- Ignacio Fernández-Tobías, Paolo Tomeo, Iván Cantador, Tommaso Di Noia, Eugenio Di Sciascio. Accuracy and Diversity in Cross-domain Recommendations for Cold-start Users with Positive-only Feedback. Proceedings of the 10th ACM Conference on Recommender Systems (RecSys 2016), Boston, Massachusetts, USA, September 2016.

- Phoung T. Nguyen, Paolo Tomeo, Tommaso Di Noia, and Eugenio. Di Sciascio. Content-based Recommendations via DBpedia and Freebase: a Case Study in the Music Domain. In 14th International Semantic Web Conference (ISWC 2015), Bethlehem, Pennsylvania, USA, October 2015.

- Tommaso Di Noia, Vito Claudio Ostuni, Jessica Rosati, Paolo Tomeo, Eugenio Di Sciascio. An analysis of users' propensity toward diversity in recommendations. Proceedings of the 8th ACM Conference on Recommender Systems (RecSys 2014), Foster City, Silicon Valley, California, USA, October 2014.

National Conferences

- Paolo Tomeo, Ignacio Fernández-Tobías, Tommaso Di Noia, Iván Cantador. Exploiting Linked Open Data in Cold-start Recommendations with Positive-only Feedback. In 4th Spanish Conference in Information Retrieval (CERI 2016), Granada, Spain, 2016.

International and National Workshops

- Paolo Tomeo, Tommaso Di Noia, Marco de Gemmis, Pasquale Lops, Giovanni Semeraro and Eugenio Di Sciascio. Exploiting Regression Trees as User Models for Intent-Aware Multi-attribute Diversity. In 2nd Workshop on New Trends in Content-Based Recommender Systems (CBRecSys 2015). 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 2015.

- Phoung T. Nguyen, Paolo Tomeo, Tommaso Di Noia, and Eugenio. Di Sciascio. An Evaluation of SimRank and Personalized PageRank to Build a Recommender System for the Web of Data. In 7th International Workshop on Web Intelligence & Communities (WIC 2015). 24th International World Wide Web Conference (WWW 2015), Florence, Italy, May 2015.

- Tommaso Di Noia, Vito Claudio Ostuni, Jessica Rosati, Paolo Tomeo, Eugenio Di Sciascio. Adaptive Diversity in Recommender Systems. Proceedings of the 6th Italian Information Retrieval Workshop (IIR 2015), Cagliari, Italy, May, 2015.

Chapter 2

Recommender Systems

2.1 Overview of Recommender Systems

Recommender Systems (RSs) are intelligent software applications that provide users with personalized suggestions about items they might be most likely interested in [128]. The term "item" indicates what the system can recommend to users: music to listen, videos to watch, products to buy, to cite a few. RSs are primarily designed to support the users in the choice of the next items to consume among a potentially overwhelming number of alternatives. Typically, recommendations are grouped in form of a ranked list of items. To perform the ranking, RSs usually try to predict the items relevance considering the user's preferences derived from her past behavior. Such preferences can be either explicitly expressed by the users (for instance by means of ratings or likes) or implicitly inferred by the user actions (for instance by mining text reviews, query searches, clicks, purchasing records or even mouse movements).

Recommendation approaches are generally classified into three main cat-

egories:

- **Content-Based** systems recommend items similar to the ones the user liked in the past, where the similarity is computed considering the item descriptive *content* - for instance the genre information.
- **Collaborative Filtering** systems take into account the past behavior of the user and suggests items liked by other users with similar tastes, where the similarity is calculated based on the similarity in the rating history of the users.
- **Hybrid** approaches combine collaborative filtering and content-based algorithms in the attempt to mitigate the weaknesses of both.

Among the different formalizations of the recommendation problem proposed in literature, the one in [9] is the most cited. The set of users is indicated as U and the set of items as I. The symbol $\mathcal{R}$ instead indicates the ratings matrix. Let $U = \{u_1, ..., u_n\}$ be the set of all the users of the system, and let $I = \{i_1, ..., i_m\}$ be the catalog of all the items. Then $\mathcal{R}$ represents the $n \times m$ utility matrix. Therefore, $r(u, i)$ indicates the rating of the user u for the item i. The recommendation task can be generally characterized as the estimate of the rating $r(u, i)$ through a recommendation function $r^* : U \times I \to \mathbb{R}$. Therefore, the problem consists in finding for each user $u \in U$ such item $i^{max,u} \in I$ maximizing the utility function r^*. More formally:

$$\forall u \in U,\ i^{max,u} = \arg\max_{i \in I} r^*(u, i)$$

Generally speaking, the user profile is basically represented by the set of feedbacks, also known as transactions [128], between the user and the RS. Such feedbacks can be either **explicitly** stated by the users, such as the rating or the like for the selected item, or **implicitly** acquired, by means of mining text reviews, query searches, clicks, time spent on web pages, or purchasing records, to cite a few. Explicit numerical ratings, such as the 15 stars used in several web site and e-commerce, are the most popular form

of feedbacks. However other forms of explicit feedbacks, such as the binary (e.g. likes/dislike) and unary (positive-only) ones, are becoming popular, due to the increase of use of social networks and content sharing websites. Implicit feedbacks represent the users' interests inferred by their actions, and are particularly useful when explicit feedbacks are not available. They can be also used to supplement the explicit feedbacks in order to provide better recommendations.

2.2 Collaborative Filtering Recommendation

In the simplest version of a Collaborative Filtering (CF) method, the recommendations for a target user are computed considering the items liked by other users with similar tastes. The similarity between two users is calculated based on the similarity in the rating history of the users. Therefore, they do not need descriptive information about the items and hence result independent from the domain in which they are applied. CF methods are divided in two main categories: neighborhood-based and model-based approaches. The former use directly the ratings to compute the predictions, while the latter use the ratings to learn a predictive model. The main advantage of neighborhood-based approaches is that they do not require any preliminary model building phase, since to compute the prediction they aggregate the ratings of the closest neighbors. Conversely, model-based techniques first learn a predictive model which is eventually used in the prediction phase.

Neighborhood-based CF

Neighborhood-based (memory-based or heuristic-based) CF methods use directly the user-item ratings matrix to compute the predictions and result very popular due to their simplicity and efficiency, besides the ability to produce accurate recommendations [106]. Such methods are divided in two categories: user-based and item-based. Given a target user, user-based systems determinate her interest for an item using the ratings given to that item

by other users, called neighbors, with similar rating patterns. Two users are considered similar if they have rated several items in the same way. Conversely, item-based approaches predict the interest of an item considering the ratings of the user for similar items. Two items are considered similar if a significant number of users have rated them in a similar way. In other words, item-based approaches aim to identify relationships between the items, useful to indirectly compute the recommendations for the users [136].

User-based neighborhood methods predict the rating $r(u, i)$ of a user u for a new item i using the ratings given to i by users most similar to u, called nearest-neighbors. Considering that the users are represented by vectors in the user-item matrix, the similarity between them can be computed by means of similarity metrics, such as Cosine Similarity or Pearson Correlation Coefficient. Relying only on the matrix of user-item interactions $\mathcal{R}$, the similarity between two users u and v, according to Pearson coefficient, is given by:

$$sim(u, v) = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i \in I}(r_{v,i} - \bar{r}_v)^2}} \tag{2.1}$$

where $\bar{r}_u$ is the average rating given by user u across all the items already evaluated. It is worth to notice that the subtraction of average values allows a fair comparison between users, making them comparable even if their rating scales are different.

After having defined the set of similar users $\mathcal{N}$ for the current user u, the prediction for the rating of user u on a new item i is given by:

$$r^*(u, i) = \bar{r}_u + \frac{\sum_{v \in \mathcal{N}} sim(u, v)(r_{v,i} - \bar{r}_v)}{\sum_{v \in \mathcal{N}} sim(u, v)} \tag{2.2}$$

Considering all the other users in the neighborhood would be both ineffective and inefficient and it is hence useful to fix the number of neighbors to a number K or to use a minimal similarity threshold [40]. The first case is known as *user-based K-Nearest Neighbor* method, often shortened to UserKNN or UNN.

The bottleneck of UNN is the search of neighbors among a large population of users, that makes infeasible to generate accurate recommendations in real-time. *Item-based nearest neighbor* methods have been proposed to address this problem [137]. Since the relationships between items are relatively static, allowing to produce accurate recommendations with offline preprocessing, while the users profiles may vary rapidly, requiring frequent and expensive updates. The idea is that items similar to the ones already chosen by the user are good candidate for recommendation. Therefore, a similarity measure sim between items is introduced and the rating of user u for item i is estimated as follows:

$$r^*(u,i) = \frac{\sum_{j \in N(i) \cap r(u)} sim(i,j) \cdot r(u,j)}{\sum_{j \in N(i) \cap r(u)} sim(i,j)}$$

where $r(u)$ represents the items rated by the user u, and $r(u,j)$ the rating value given by the user u with respect to the item i. Therefore, the above equation takes into account the neighbors of i belonging to the user profile and computes an average of the user ratings to such neighbors weighted by the similarity values.

As in the user-based approach, the size of the neighborhood is generally restricted to a specific size, resulting in the *item-based K-Nearest Neighbor* method, often shortened to ItemKNN or INN. The cosine similarity measure is a standard metric for estimating how similar items are. After having represented items i_1 and i_2 as vectors, indicated respectively with $\boldsymbol{i_1}$ and $\boldsymbol{i_2}$, the cosine similarity corresponds to the cosine of the angle between $\boldsymbol{i_1}$ and $\boldsymbol{i_2}$ and, specifically, is given by:

$$sim(i_1, i_2) = \frac{(\boldsymbol{i_1})^T \boldsymbol{i_2}}{||\boldsymbol{i_1}||\ ||\boldsymbol{i_2}||} \tag{2.3}$$

where $||\boldsymbol{i_j}||$ is the Euclidean norm of vector $\boldsymbol{i_j}$.

Therefore, the similarity function is defined as follows

$$sim(i_1, i_2) = \frac{\sum_{u \in U}(r_{u,i_1} - \bar{r}_{i_1})(r_{u,i_2} - \bar{r}_{i_2})}{\sqrt{\sum_{u \in U}(r_{u,i_1} - \bar{r}_{i_1})^2}\sqrt{\sum_{u \in U}(r_{u,i_2} - \bar{r}_{i_2})^2}} \tag{2.4}$$

where U, in this case, is restricted to the set of users who rated both items i_1 and i_2.

An *adjusted* version of cosine similarity measure is used to take into account the differences in rating scale among the users [136]. Specifically, this version subtracts the corresponding user average from each rating. More formally:

$$sim(i_1, i_2) = \frac{\sum_{u \in U}(r_{u,i_1} - \bar{r}_u)(r_{u,i_2} - \bar{r}_u)}{\sqrt{\sum_{u \in U}(r_{u,i_1} - \bar{r}_u)^2}\sqrt{\sum_{u \in U}(r_{u,i_2} - \bar{r}_u)^2}} \tag{2.5}$$

It is important to note that **item-based neighborhood** approaches are placed among collaborative methods only if the vectorial representation is based on information drawn from users' ratings.

Model-based CF

In contrast to neighborhood-based systems, which use directly the user-item matrix in the prediction computation, model-based approaches learn a predictive model from that matrix. Then, the trained model can be used to compute recommendations for individual users. The learned model parameters represent salient characteristics, called **latent factors**, of users and items inferred from the observed interactions patterns between the users and items [82].

These methods have been introduced to overcome the problems of scalability of other methods, since data are preprocessed offline and the only the learned model is used to make recommendations. **Matrix Factorization** (MF) models represent some of the most successful realizations of model-based approaches, combining good scalability and predictive accuracy [82]. Basically, they learn low-rank representations of the user-item matrix, deriving from the rating patterns some latent factors and then map both users and items to a joint latent factor space of dimensionality k, where k is usually much smaller then the size of the original user-item matrix. Therefore, the interactions between users and items are modeled as inner products in the latent factor space. More formally, each item i is associated with a vector

$\boldsymbol{q_i} \in \mathbb{R}^k$, where each element in q_i indicate the extent to which the item possesses the corresponding factor. While each user u is associated with a vector $\boldsymbol{p_u} \in \mathbb{R}^k$, whose elements indicate the importance of the corresponding factor for the user. Finally, their scalar product captures the interaction between user u and item i and can be used for rating estimation as follows

$$r^*(u,i) = \boldsymbol{q_i^T p_u} \tag{2.6}$$

While the rating can be easily estimated once the model is computed, the major challenge remains to identify accurate mappings. Earliest implementations of matrix factorization methods relied on imputation techniques to remove the user-item matrix sparsity filling in the void cells, for instance with the average ratings for a user or for an item [138]. However, imputation increases the amount of data, making the computation more expansive, and moreover can lead to less accurate recommendations [82]. Recently, a number of MF methods that model directly the observed ratings only have been proposed. In the most common formulation of MF, to learn the model (i.e. q_i and p_u vectors) the systems aims at optimize a regularized squared error cost function, as follows

$$\min_{q^*,p^*} \sum_{(u,i)inK} (r(u,i) - r^*(u,i) + \lambda \left(\sum_u ||\boldsymbol{q_i}||^2 + \sum_i ||\boldsymbol{p_u}||^2 \right) \tag{2.7}$$

where K is the set of the (u,i) pairs representing all the known interactions in the user-item matrix, and the term λ controls the importance of the regularization term used to prevent overfitting. The choice of λ depends on the data and can be determined by cross validation. Usually, *Stochastic Gradient Descent* is used for minimizing that optimization problem and hence learning the factors, e.g. [138, 59, 56]. As the equation 2.7 is not convex in both q_* and p_*, *Alternative Least Squares* can be used to make the problem quadric. ALS is based on the observation that when all the parameters but one are fixed, the cost function 2.7 becomes a standard least-squares

problem and then the solution can be optimally computed. ALS is particularly favourable to massively parallelize the learning algorithm, since it can compute q_i and p_u vectors independently of the other vectors [82].

A specific matrix factorization method has been proposed in [67] for better handling implicit feedback, taking into account both observed and unobserved feedback in the training process. The motivation is that the model should not only be able to predict high scores for relevant items, but also whether an item was rated or not. However, non-observed user-item interactions can be due to the user not liking the item or not knowing about the item. Hence, the model includes a confidence hyperparameter $c_{(u,i)}$ in the loss function to penalize mistakes on observed and non-observed preference predictions differently

$$\min_{q^*,p^*} \sum_{(u,i)} c_{u,i}(p(u,i) - r^*(u,i))^2 + \lambda \left(\sum_u ||\boldsymbol{q_i}||^2 + \sum_i ||\boldsymbol{p_u}||^2 \right) \tag{2.8}$$

Where $p(u,i)$ represents a binarized derivation of $r(u,i)$, as follows

$$p(u,i) = \begin{cases} 1 & \text{r(u, i)} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.9}$$

The confidence parameter is set $c_{(u,i)} = 1 + \alpha r(u,i)$ with $\alpha > 0$, so that mistakes predicting observed feedback are more penalized. It is important to note that equation 2.8 considers all the possible (u,i) pairs, while the equation 2.7 uses only the known interactions in the user-item matrix. It is possible since $p(u,i)$ assumes 0 for all the unknown (u,i) pairs. The intuition behind this method is that the systems has minimal confidence in $p(u,i)$ for every user-item pair, but as it observes more evidence for positive preference, the confidence in $p(u,i)$ increases accordingly [67].

Even though collaborative filtering is the most widely adopted approach, it suffers from some drawbacks. Firstly, it strongly depends on the amount of rating information, and struggles with finding accurate recommendation in case of high level of *data sparsity*, or in *cold-start* situations. A cold-start

situation exists when a recommender system has no enough historical information about a user or an item [49]. In turn, the inability of this methods to handle the item cold-start problem leads to less novel and diverse recommendations and can foster the *rich-get-richer* effect for popular products [5]. Another problem that affects CF methods is the *grey sheep problem*, that represents the inability of the system to treat users with unusual preferences.

2.3 Content-based Recommendation

Content-based methods rely on the assumption that a user likes items similar to the ones she likes in the past, where the similarity is determined on their descriptive characteristics, that are usually domain-dependent. For example, in the movie domain each movie can be represented by means of its genres, the title, the name of the director, the plot, and so on. Then, this kind of methods represent the user profile as an assignment of importance to those features, converting the recommendation task into a matching between items characteristics and user's preferences. Given a candidate item and a target user, the system tries to determinate the level of her interest in that item.

Generally, such characteristics are extracted from metadata associated to the items, or directly from the textual description. In the former case, the content extracted could be not enough to describe the items, due to the scarcity and/or low quality of metadata information. While in the latter case, the natural language ambiguity in the textual description involves several complications in the extraction task. This problem is known as *limited content analysis*, and could lead to less discriminative characterizations and hence final recommendations of poor quality. The use of semantic technologies and open knowledge sources, such as Wikipedia, DBpedia and Wikidata, to face the limited content analysis problem represents one of the most innovative and promising line of research [39].

Another limitation of content-based methods regards the content overspecialization, namely the inability to suggest relevant items but also different

to the ones the user already knows, penalizing the recommendation novelty and in turn the discovery of new content.

2.4 Hybrid Recommendation

The main idea behind hybrid recommender systems is to combine collaborative and content-based algorithms in order to mitigate the weaknesses of the individual approaches and obtain better recommendation quality. For instance, CF methods are not able to recommend new items, while content-based methods do not suffer from this problem. To date, different hybridization methods have been proposed and a comprehensive taxonomy is given in [25]:

- **Weighted**: the scores of the different recommender systems are combined by weighted sums;
- **Switching**: the system chooses one among different methods and turns all the others off, depending on the user profile or recommendation quality of the different systems;
- **Mixed**: recommendations generated by several recommenders are combined together in the same list by means of a ranking or combination strategy;
- **Feature combination**: the features derived from different knowledge sources (e.g. collaborative and content features) are used together by the same recommender system;
- **Cascade**: the recommendation task relied on a pipeline process where each recommender refines the recommendations given by the previous one;
- **Feature augmentation**: the output of a recommender augments the feature space of the subsequent recommender;

- **Meta-level**: the model generated by one recommender is used as input by a principal recommender.

Two types of methods based on feature combination are of particular interest for this thesis: collaborative filtering with side information, and graph-based approaches.

2.5 Collaborative Filtering with Side Information

Side information about users and items can be exploited to improve the recommendation quality of collaborative filtering methods, in particular in cold-start scenarios [49]. In one of the first approach proposed to inject content information in a CF method, missing values in the user-items matrix are predicted using items side information and then a classical user-based CF is applied on the dense matrix [99]. To date, more sophisticated methods have been proposed in this area [4, 36, 117, 107, 120].

Collective Matrix Factorization (CMF) [147] is a representative method of this approach that originally showed significant improvements when item genres are taken into account for computing movie recommendations. The idea behind CMF is to simultaneously factorize the user-item matrix and the item-item similarity matrix. Predictions are still computed using Equation 2.6, but CMF includes an additional set of item latent vectors $s_j \in \mathbb{R}^K$ feature to model the pairwise item interactions through the similarities. The loss function becomes:

$$\min_{q^*,p^*,r^*} \gamma \sum_{(u,i)} c_{u,i}(p(u,i) - r^*(u,i))^2 + (1-\gamma) \sum_i \sum_j (s_{i,j} - \boldsymbol{q_i^T s_j})^2 + \lambda \left(\sum_u ||\boldsymbol{q_i}||^2 + \sum_i ||\boldsymbol{p_u}||^2 + \sum_j ||\boldsymbol{s_j}||^2 \right) \tag{2.10}$$

where $s_{i,j}$ represents the content-based similarity between items i and j; $\gamma \in (0,1]$ weights the importance of the item similarities in the factorization.

If $\gamma = 1$ the result is the same of IMF, whereas γ close to 0 would ignore the preference predictions.

Factorization Machines (FMs) [120] are becoming increasingly popular, as they provide a principled and generic approach to integrate metadata into MF, showing promising results in the task of context-aware recommendation. Finally, a different set of approaches jointly factorize the user-item preference and item-metadata matrices, sharing the item latent factors between both decompositions. Therefore, Factorization Machines provide a generic way to extend the standard MF model with different kinds of side information. The idea is to (one-hot) encode the user-item-metadata information in a single feature vector $x \in \mathbb{R}^{n=|U|+|I|+|F|}$ where $|U|, |I|, |F|$ are the number of users, items, and features, respectively. The model equation for a factorization machine of degree d = 2 is defined as:

$$r^*(u,i) = w_0 + \sum_{a=1}^{n} w_a x_a + \sum_{a=1}^{n} \sum_{b=a+1}^{n} \langle \boldsymbol{v_a}, \boldsymbol{v_b} \rangle \, x_a x_b \tag{2.11}$$

where the model parameters w_0, $\boldsymbol{w} \in \mathbb{R}^n$, $\boldsymbol{V} \in \mathbb{R}^{n \times k}$ have to be estimated, and $\langle \cdot, \cdot \rangle$ indicates the scalar product of two vectors. A row v_i in V represents the i-th variable with k factors.

The w_a parameters model the contribution of each component in the feature vector, whereas the weights for the pairwise interactions are factorized as the product of two latent feature vectors v_a and v_b. Factorization machines generalize IMF by also taking into account user-feature and item-feature interactions, which are also factorized. It has been demonstrated that the model equation 2.11 can be computed in linear time, therefore the models parameters can be learned efficiently by gradient descent methods [121]. Summing up, FMs can successfully incorporate additional information sources associated with users and items at a low computational cost [123].

2.6 Graph-based Recommender Systems

The importance of graph-based approaches to recommendation has emerged concurrently with the increasing availability of additional user and item information useful for the recommendation process itself. These approaches allow combining the user-item rating matrix with side information into a graph, and then applying a graph mining technique. More specifically, as shown in Figure 2.1, the rating matrix is transformed into a bipartite graph component which consists of user and item nodes linked with rating/like edges extended to form a multipartite graph, including nodes representing additional entities, which are related to items. The graph also allows including other edges, representing e.g. contextual information for the ratings, social connections between users, and semantic relations between entities [145]. The result thus can be defined as a heterogeneous information network consisting of a multi-typed and multi-relational directed graph, with nodes and edges of different types [151].

2.6.1 Heterogeneous Information Network

Structuring all the available data in form of a graph leads to different advantages: (i) well-known graph-based algorithms can be used to develop hybrid recommender systems able to exploit the different types of information surfing the graph [171]; (ii) both content and collaborative aspects are represented in a uniform setting thus leveraging the multi-relational nature of the graph; (iii) the graph can be directly extended with information already available in the form of graphs, such as Linked Data [108]; (iv) exploring the graph jumping different hops could produce relevant but not obvious recommendations and also help on addressing the cold-start scenario, since exploring longer paths in the network could overcome the lack of connection information between users and items. PathRank [85] is an extension of the Personalized PageRank algorithm able to exploit different paths on a heterogeneous graph during the random walk process. At each iteration

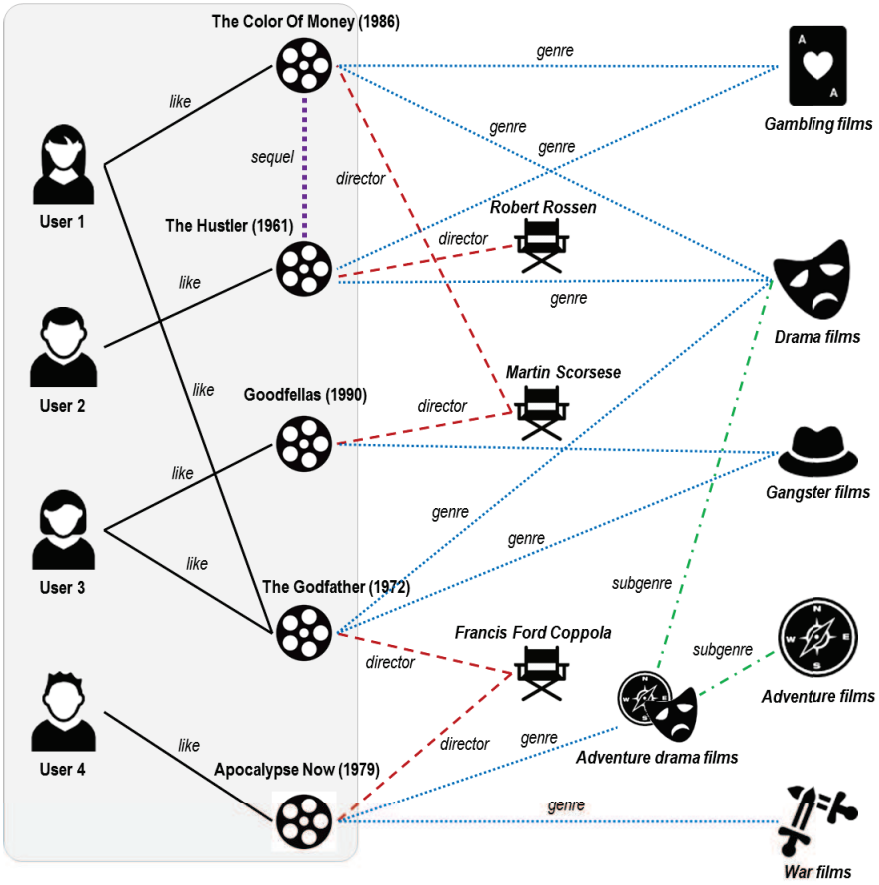


Figure 2.1: Example of heterogeneous information network

the random walker has three options: transition, move to one of adjacent nodes; restart, restart the random walk from one of the query nodes; path following, considering one of meta-paths that the authors call path-guides. A meta-path is a path consisting of a sequence of typed relations. HeteRec [171] is a hybrid method based on matrix factorization that uses meta-path based latent features to represent the connectivity between users and items along different types of paths in a heterogeneous information network. HeteRec defines a user preference diffusion score extending the meta-path based similarity PathSim [151], including the user implicit feedback. This process propagates user preferences along the different meta-paths in the graph, producing a user-item matrix for each meta-path where each cell indicates the

probability of certain user reaches a certain item under the relative meta-path. Then, it factorizes each matrix, and builds a recommendation model that estimates the rating for a user-item pair computing a weighted sum of the relative latent features in the matrices.

A more formal overview of the aforementioned methods is described below. Given a graph G, our aim is to produce personalized recommendations leveraging the knowledge encoded in the graph. As described in Section 2.6.1, data derived from different knowledge sources can be combined by means of heterogeneous information network, which consists in a graph with different types of nodes and relations. Therefore, it is possible to find different paths among users and items composed by different types of relations. For example, an user may be connected to an item i by the relation (like, director, director^{-1})[1], which basically means that the user likes one or more items with same director of item i. More formally, these paths are called meta-paths and an actual sequence of nodes and relations, which generates the particular path, is called path instance [151].

2.6.2 HeteRec

HeteRec is a graph-based recommender system based on an adaptation of HeteRec (Section 2.1). Briefly, it computes for each meta-path the relative diffused user preferences matrix extending the similarity measure PathSim [151] in order to include the user feedbacks. More formally, the user preference diffusion score between user u and item j, along a generic meta-path P, is defined as:

$$sim(u, j) = \sum_{i \in R(U)} \frac{2 \cdot r(u, i) \cdot |p_{i \to j} : p_{i \to j} :\in P|}{|p_{i \to i} : p_{i \to i} \in P| + |p_{j \to j} : p_{j \to j} \in P|} \tag{2.12}$$

where $p_{x \to y}$ is a path instance between the items x and y. Basically, this formula is a weighted sum of PathSim among the items in the user profiles and

[1] Given a relation r going from x to y we can denote with r^{-1} the relation going from y to x.

the target item j, where the numerator measures the connectivity defined by the number of path instances between them following P and the denominator represents the balance of their popularity in the graph, namely the number of path instances between themselves. Once the matrices are computed, HeteRec factorizes them with a low-rank matrix factorization technique. could remove valuable information in the cold start scenario. Therefore, our model is directly based on the not factorized diffused user preferences matrices. Finally, the estimated user-item preference matrix is computed as the weighted sum of the different meta-path matrices: $R^* = w_{P_1} \cdot R^*_{P_1} + ... + w_{P_m} \cdot R^*_{P_m}$, where m is the number of meta-paths, w_{P_1} and $R^*_{P_1}$, respectively, the weight and the diffused user preferences matrix of i-th meta-path. HeteRec splits the users into clusters, and then computes the importance of each meta-paths with a learning-to-rank approach. However, in user cold-start situation, clustering the users is impracticable with a few ratings and without additional information. Therefore, meta-paths weights should be computed globally for all the cold-start users.

2.6.3 PathRank

PathRank, a graph-based algorithm presented in [85], extends the Personalized PageRank considering the connectivity between users and items along different meta-paths. At each iteration, the random walker has three options: *transition*, move to one of adjacent nodes with probability w_{trans} ; *restart*, restart the random walk from one of the query nodes with probability $w_{restart}$; path following, considering one of the meta-paths with probability w_{path}. Therefore the PathRank vector $\vec{r}$ is computed as:

$$\vec{r} = w_{trans} M_G^T \vec{r} + w_{restart} \vec{t} + w_{path}(w_{P_1} M_{P_1}^T + ... + w_{P_m} M_{P_m}^T)\vec{r} \qquad (2.13)$$

where M_G is the item-item transition matrix of the full graph G, M_{P_1} is the transition matrix of the i-th meta-path, $\vec{t}$ is is the teleport vector representing the recommendation query (user profile) initialized with $1/|R(u)|$ for

each item in $R(u)$, 0 otherwise.

2.7 Linked Data for Recommender Systems

In the last year we are witnessing the evolution of the Web we used to know towards a huge distributed knowledge base. The *World Wide Web* is moving fast from a network of documents to a network of interconnected data (entities) thus creating the so called `Web of Data`. This latter has been introduced as a new scheme for bringing structured data into the Web. Data in the `Web of Data` is represented as a graph of semantically connected resources by means of `RDF` thus allowing the structured data provided by `Linked Data` to be not only graspable by human beings but also processable by computers. This paves the way for automatic processing of Web contents, thus helping to mine existing data and deduce new knowledge. To date, information sharing, information retrieval [54, 55], community detection, recommendation systems [41, 115, 108] - to name a few - are the noteworthy applications that successfully leverage `Linked Data`. Having high quality, well-structured information about the items to recommend, it is possible to design semantic-aware recommender systems that can provide better suggestions and deal with important quality factors such as diversity and novelty. Some notions on RDF and SPARQL, and a brief description of DBpedia are showed in Appendix A.

2.7.1 Feeding Recommender Systems with LD

In order to feed recommender systems with `LD`, the first step is to link the items in the system with the corresponding resources in the `LD` knowledge bases. There are two main ways for performing this linking task: Direct Item Linking and Item Description Linking. The former is the more straightforward way, but requires that the items have the corresponding `LD` resources. For instance, in the movie domain, using title and year of each movie it is possible to find the relative DBpedia resources. While, the Item Descrip-

tion Linking method links metadata or textual descriptions to LD resources. Such information can be used as input for entity linking tools in order to have access to LD resources and link them to the item. Specifically, Entity Linking is the task of linking the entity mentioned in the text with the corresponding real world entity in the existing knowledge base [144]. Many Entity Linking tools have been proposed in the literature and made available on the Web. Some of them are: Babelfy [102], Dexter [31], DBpedia Spotlight [100], TAGME [52], NERD [129].

Once the items are linked to a LD source, extracting information for each item from the knowledge base is an easy task. By means of SPARQL queries, descriptive and informative subgraph can be extracted in form of set of RDF triples. Eventually, all the extracted portions of LD can be merged to obtain a specific knowledge graph representative of the domain of interest covered by the recommender. However, finding a small enough - both informative and compact - subgraph descriptive of the item is not a easy task. Considering that data in LD sources are represented in an ontological structure, selecting classes and properties is possible to reduce the dimension of the extracted graph and, at the same time, finding the most informative information for a specific task. Therefore, methods for feature selection have been proposed for selecting the most relevant properties for recommendation tasks [119, 103]. Feature selection is a process to automatically select the attributes in a dataset that are most relevant to the predictive model at hand. It is useful to remove irrelevant or redundant attributes that do not contribute to the accuracy of the predictive model or that can, indeed, decrease the accuracy of the model itself.

2.7.2 SPrank

SPrank (Semantic Path-based ranking) [109, 108] is a hybrid recommendation algorithm able to combine ontological knowledge belonging to the Web of Data with collaborative information in a heterogeneous information network in a learning to rank setting. Learning to rank is a task to automatically

construct a model for ranking new objects according to their degrees of relevance, or preference [90]. To compute the *top-N* recommendations, the standard learning to rank setting adopted in Information Retrieval [90] can be adapted by replacing queries with users and documents with items and using user's ratings as relevance scores. This allows the use of well established learning to rank techniques [27, 75]. For each user-item interaction (u, i), SPrank encodes the matching between user interests and item content in the feature vector $\mathbf{x_{ui}} \in \mathbb{R}^D$ where D is the dimension of the feature space. Each component in $\mathbf{x_{ui}}$ represents the relevance of i for u with respect to a specific feature[2]. Therefore, the goal of SPrank is to learn a ranking function $f : U \times I \to \mathbb{R}$ from the training data able to approximate for each user its ideal ranking, by means of learning to rank approach. Eventually, SPrank learns a single model over all users, where the matching user's and item's characteristics is explicitly represented using the joint user-item feature vector $\mathbf{x_{ui}}$.

For each user $u \in U$ we define user profile $I_u = \{i \in I | r_{ui} \in R\}$ the set of items rated by u. The ratings r_{ui} associated to the items in I_u can be used to induce a ranking among them. Defining the training set TS and recommendation set RS as follows:

$$TS = \bigcup_u \{\langle u, i, \mathbf{x_{ui}}, r_{ui}\rangle | i \in I_u\}$$

$$RS = \bigcup_u \{\langle u, i, \mathbf{x_{ui}}, r^*_{ui}\rangle | i \in (I \setminus I_u)\}$$

The goal is then to learn a scoring function $f : U \times I \to \mathbb{R}$ from the training data TS able to replicate for each user its perfect ranking. Once we have $f(u, i)$ we apply it to the recommendation set RS for composing the *top-N* recommendation list by means of the computed scores r^*_{ui}.

[2] SPrank relies on *path-based* features as detailed in Section 2.7.2.

Learning with Implicit Feedback Data

The formulation of the learning problem given above relies on the availability of graded or binary relevance values commonly obtained from explicit feedback data. However in many real scenarios explicit user feedback are difficult to obtain. Nevertheless, it is still possible to obtain implicit feedback data by analysing users interactions with the system. Examples of such interactions can be clicks, purchases, video watching, etc. The main problem with implicit feedback is that they reflect only positive user preferences. If a user buys an item it is reasonable to assume that the user likes it. On the contrary the system cannot infer anything about what the user dislikes. The unobserved data are a mixture of actually negative and missing values [122], but the system does not have any information for discriminating between them. Then, as all items have same (unary) relevance, the learning task becomes infeasible. To face this problem, we select a portion of unobserved items $I_u^* \subset I \setminus I_u$ for each user to be used as negative data points in the training set which becomes:

$$TS = \bigcup_u \{\langle u, i, \mathbf{x_{ui}}, r_{ui}\rangle | i \in (I_u \cup I_u^*)\}$$

where $r_{ui} = 0$ for each $i \in I_u^*$.

Path-based features

SPrank extracts features characterizing the interactions between users, items and entities capturing the complex *paths* between them. Due to the multi-relational nature of the data in LD sources, there are several types of paths. Each particular path has its own semantics and it might have a different relevance for the end user. For instance a user might be interested in a movie because of few specific actors and in such case paths involving the `starring` relation would be more discriminative in finding good movies. Another user might not be really interested in few specific actors but rather in those actors belonging to a specific category. In this case considering a sequence of

Table 2.1: Example of a Path Index in SPrank.

	Path Index
1:	(like, subsequentWork^{-1})
2:	(like, dislike^{-1}, like)
3:	(like, like^{-1}, dislike)
4:	(dislike, like^{-1}, like)
5:	(dislike, like^{-1}, like, subsequentWork^{-1})
6:	(like, dislike^{-1}, like, subsequentWork)
7:	(like, like^{-1}, dislike, like^{-1}, like)
8:	(like, literaryGenre, literaryGenre^{-1})
9:	(like, author, author^{-1})
10:	(like, subject, broader, broader^{-1}, subject^{-1})
11:	(dislike, subject, broader, broader^{-1}, subject^{-1})
12:	(like, notableWork, InfluencedBy, subject, subject^{-1}, author)
13:	(like, author, InfluencedBy, subject, subject^{-1}, author)
14:	(like, author, subject, subject^{-1}, InfluencedBy^{-1}, author^{-1})
15:	(like, author, subject, subject^{-1}, InfluencedBy^{-1}, notableWork^{-1})
16:	(dislike, notableWork, InfluencedBy, subject, subject^{-1}, author)
17:	(dislike, author, InfluencedBy, subject, subject^{-1}, author)
18:	(like, subsequentWork^{-1}, subject, broader, broader^{-1}, subject^{-1})
19:	(like, subject, broader, broader^{-1}, subject^{-1}, subsequentWork)
20:	(like, subsequentWork, like^{-1}, like)

relations such as `(starring, subject)` would be better.
The intuition behind SPrank is that rich features based on sequence of relations can be extracted for building the feature vector $\mathbf{x_{ui}}$ and then a learning to rank algorithm can be used for discerning what paths are fmost relevant.

Given the multi-relational graph G as defined in Section 2.6.1, `path` is defined as the sequence of relations of the form $(r_1 \dots \sigma_l \dots \sigma_L)$ such that $r_1 = (u, v_1)$, $\sigma_l = (v_{l-1}, v_l)$ and $\sigma_L = (v_{L-1}, i)$ with $u \in U$, $i \in I$. A path instance refers to the actual sequence of nodes and relations $u \xrightarrow{r_1} v_1 \dots \xrightarrow{\sigma_l} \dots v_{L-1} \xrightarrow{\sigma_L} i$, and the *length of a path* is the number of relations contained within such path. Considering only paths with length greater than 1 and less or equal than a given L, $Path$ indicates the index of all the possible paths in G and $Path(j)$ denotes the j-th entry.

Considering a user-item pair (u, i), $\#path_{u,i}(j)$ denotes the number of path instances between u and i corresponding to the specific $Path(j)$ entry in the index. In other words, it represents how many paths of type $Path(j)$

connect u and i. Then, the j-th component in the feature vector $\mathbf{x_{ui}}$ is defined as follows:

$$\mathbf{x_{ui}}(j) = \frac{\#path_{u,i}(j) - \min_{k \in I} \left(\#path_{u,k}(j)\right)}{\max_{k \in I} \left(\#path_{u,k}(j)\right) - \min_{k \in I} \left(\#path_{u,k}(j)\right)} \tag{2.14}$$

Equation (2.14) represents the importance of the path $Path(j)$ between u and i in the graph involving all nodes reachable in L hops starting from u. Given the user u and considering the specific path $Path(j)$, we observe how this path is distributed among all items for that user. Since the absolute count values of a path feature for different users might not be comparable, user-based normalization is applied for each feature as proposed by [91] for query-document pairs.

2.8 Recommender Systems Evaluation

The evaluation of recommender systems is inherently difficult to perform: the same algorithm may be better or worse on different datasets, different tasks may require different evaluation goals, and the evaluation results may considerably change depending on the evaluation setting adopted. Adopting a rigorous evaluation strategy is required to obtain correct results, and providing all the details about the experiments is fundamental for comprehensibility and replicability of research results.

The evaluation can be either offline or online. In the first case, past data are used to train and test the systems. Since we are interested in the future performance on new data, we must properly partition the original dataset into training and test sets [38]. The test set must be different and independent from the training set in order to test the systems on unseen data and correctly evaluate its ability to predict future interaction between users and items. Usually, a splitting strategy is applied for each user in the dataset, dividing the profile into 80% training and 20% test data. Among different splitting strategies, random and time-based splitting ones are the most commonly used. The latter requires that more recent interactions of the users are

selected for testing and the older one are used for the training phase. Once the dataset is splitted and the recommendations are computes, performance metrics are computed comparing recommendation lists with test data. Next Section shows some well-known metrics for evaluating the recommendation accuracy. Metrics for other quality factors, such as diversity and novelty, are detailed in Section 2.9.4. The alternative is the online evaluation in which the performance of a new system is evaluated with real user feedback. Online evaluation can be in form of user studies - a number of qualitative measurements are gathered by means of surveys or supervising the users' behaviours during the interaction - or in form of A/B testing - two different versions of the system, control (A) and variation (B), are presented to two different groups of users, to determine which of the two variants is more effective.

In this work we focus on offline evaluation which allows low-cost assessment of several recommendation algorithms and configurations. Another advantage is that it allows to measure the performance in an objective manner, using metrics which can not be affected from external conditions, as in the case of online evaluation.

2.8.1 Accuracy Metrics

For evaluating the accuracy several metrics have been proposed. In this work we use $Precision@N$, $Recall@N$ and $nDCG@N$. The first one represents the fraction of relevant items in the top-N recommendations. Let $rel(u, i)$ be a boolean function that represents the relevance of item i for the user u, with value 1 for relevant and 0 for non-relevant items, then $Precision@N$ is calculated as follows

$$Precision@N = \frac{\sum_{i=1}^{N} rel(u, i)}{N} \tag{2.15}$$

$Recall@N$ indicates the fraction of relevant items, in the user test set, occurring in the top-N list. Being $test(u)$ the set of relevant items in the test

set for the user u, $Recall@N$ is defined as

$$Recall@N = \frac{\sum_{i=1}^{N} rel(u,i)}{\mid test(u) \mid} \tag{2.16}$$

Although precision and recall are good indicators to evaluate the accuracy of a recommendation engine, they are not rank-sensitive. On the other side, $nDCG@N$ takes into account the position of a relevant item in the recommendation list. More formally

$$nDCG@N = \frac{1}{\text{iDCG}} \cdot \sum_{i=1}^{N} \frac{2^{rel(u,i)} - 1}{log_2(1+i)} \tag{2.17}$$

where iDCG is a normalization factor that sets nDCG@N value to 1 when an ideal ranking is returned [17].

2.9 Diversity in Recommender Systems

Recently, the drawbacks of building recommendation engines focusing exclusively on accuracy maximization have been explored and highlighted [2, 23, 98]. As recommendations are usually presented in form of list or group, the user experience strongly depends on the overall quality of such recommendations, thus the diversity among them results one of the most important quality factor [98, 29]. Simply put, the most accurate recommendations for a user are often too similar with each other (e.g., songs by the same artist), or *overspecialized*, thus causing user dissatisfaction and frustration [175]. The so called *portfolio effect* in recommender systems [24] has been widely recognized as a situation when very similar, almost identical, items appear in a recommendation list [152], correctly but bothering the user [178] (see Figure 2.2). The need to move beyond traditional accuracy metrics in the evaluation of a recommendation engine has been originally argued in [148] and several works have tackled the issue of diversification of recommendations as a way to increase user's utility [23, 16, 168, 156, 21], reaching the conclusion that a degree of diversity in the list can be increased at a cost of reducing

system accuracy [29]. It is noteworthy that the relation between accuracy and diversity goes beyond a mere trade-off as recently pointed out in [46], where the authors provide an analysis of user's perception of differences in recommendation algorithms and show that there is a strong correlation between perceived accuracy and satisfaction of the users for algorithms able to better diversify the returned list of recommended items.

(a)

(b)

Figure 2.2: Example of two recommendation lists where (a) contains all movies with same genre (Action), while (b) has higher diversity in terms of genre.

Generally, accuracy and diversity are considered as contrasting properties, due to the demonstrated trade-off between them in offline evaluation [29]. In spite of that, a recent user study proved that diversity in recommendations has an important positive impact on the user satisfaction [46] and another research demonstrated that providing a personalized degree of diversification, taking into account the users propensity towards diversity,

may foster the recommendation diversity without affecting accuracy or even slightly improve it [43]. Interestingly, the diversity issue has been primarily considered in the Information Retrieval field. As user queries are often ambiguous and the intent is hence not clear, proposing a set of answers covering different intents may increase the probability that users find at least one relevant document [135, 33]. Then, the concept of intent-aware diversification has been applied in the Recommender Systems field [162], where user interests with reference to items characteristics correspond to user intents in information retrieval, and extensively studied thus producing new algorithms and evaluation metrics [159, 156, 164].

There is a noteworthy effort by the research community in addressing the challenge of recommendation diversity. That interest arises from the necessity of avoiding monotony in recommendations and controlling the balance between accuracy and diversity, since increasing diversity inevitably puts accuracy at risk [177]. However, a user study in the movie domain [46] demonstrates that user satisfaction is positively dependent on diversity and there may not be the intrinsic trade-off when considering user perception instead of traditional accuracy metrics.

Typically, the proposed approaches aim to replace items in an already computed recommendation list, by minimizing the similarity among all items. Some approaches exploit a re-ranking phase with a greedy selection (see Section 2.9.1), for instance [156], or with other techniques such us the Swap algorithm [169], which starts with a list of K scoring items and swaps the item which contributes the least to the diversity of the entire set with the next highest scoring item among the remaining items, by controlling the drop of the overall relevance by a pre-defined upper bound.

Other types of approaches try to directly generate diversified recommendation lists. For instance, [1] proposes a probabilistic neighborhood selection in collaborative filtering for selecting diverse neighbors, while in [146], an adaptive diversification approach is based on Latent Factor Portfolio model for capturing the user interests range and the uncertainty of the user prefer-

ences by employing the variance of the learned user latent factors. In [125] it is proposed a hybrid method based on evolutionary search following the Strength Pareto approach for finding appropriate weights for the constituent algorithms with the final aim of improving accuracy, diversity and novelty balance. [176] considers the problem to improve diversity while maintaining adequate accuracy as a binary optimization problem and proposes an approach based on solving a trust region relaxation. The advantages of this approach is that it seeks to find the best sub-set of items over all possible sub-sets, while the greedy selections finds sub-optimal solutions.

Multi-attribute diversity has been substantially non-treated in the literature of recommender systems. A recent work [43] proposes an adaptive approach able to customize the degree of recommendation diversity of the *top-N* list taking into account the inclination to diversity of the user over different content-based item attributes. Specifically, entropy is employed as a measure of diversity degree within user preferences and used in conjunction with user profile dimension for calibrating the degree of diversification.

Furthermore, increasing attention has been paid to the intent-aware diversification, namely the process of increasing the diversity taking into account the user interests. Some approaches are based on adapted algorithms proposed for the same purpose in the Information Retrieval field, such as IA-Select [30] and xQuAD [135]. A recent improvement of xQuAD, proposed in [164], uses a constrained PLSA for the intent modeling task that uses explicit aspects, but learns the aspect probabilities to directly optimise their predictive performance. An approach for extraction of sub-profiles reflecting the user interests has been proposed in [159]. Such sub-profiles are used to generate recommendations for each of them, with the aim of maximizing the number of user tastes represented and simultaneously avoiding redundancy in the top-N recommendations. A new approach, proposed in [157], builds on binomial a greedy re-ranking algorithm and combines global item genre distribution statistics and personalized user interests to satisfy coverage and non-redundancy of genres in the final list.

Aggregate diversity, also known as sales diversity, is considered another important factor in recommendation for both business and user perspective: the user may receive less obvious and more personalized recommendations, complying with the target to help users discover new content [160] and the business may increase the sales [53]. [7] proposes the concept of aggregated diversity as the ability of a system to recommend across all users as many different items as possible and proposes efficient and parametrizable re-ranking techniques for improving aggregate diversity with controlled accuracy loss. Those techniques are simply based on statistical informations such us items average ratings, average predicted rating values, and so on. [160] explores the impact on aggregate diversity and novelty inverting the recommendation task, namely ranking users for items. Specifically, two approaches have been proposed: one based on an inverted neighborhood formation and the other on a probabilistic formulation for recommending users to items. [134] proposed a k-furthest neighbors collaborative filtering algorithm to mitigate the popularity bias and increase diversity, considering also other factors in user-centric evaluation, such as novelty, serendipity, obviousness and usefulness.

2.9.1 Greedy Selection Algorithms

The activity of a recommender system can be divided into two phases: first there is the prediction of the ratings for unrated items and then the items can be re-ranked to maximize user's utility. According to [7], the re-ranking phase can be applied to improve diversity, without modifying the recommendation process. However, finding the most diverse results is a NP-hard problem and hence several heuristics have been proposed [83]. Most previous diversification approaches are based on a greedy selection strategy [28, 135, 13]. Such strategy selects the next most relevant item only if that item is diverse with respect to the items already selected [83].

Hereafter, we will use overlined bold capital letters to denote lists, e.g., $\overline{\mathbf{X}}$, and bold capital letters to represent the corresponding set of elements belonging to the list, e.g., $\mathbf{X}$. Let $\overline{\mathbf{R}} = \langle 1, ..., n \rangle$ be the recommendation

list for user u generated using the predicted ratings and suppose we want to provide the user with the re-ranked list $\overline{\mathbf{S}}$ of recommendations, such that $\mathbf{S} \subset \mathbf{R}$ and whose length is $N \leq n$. The adopted greedy strategy can be explained through Algorithm 1. At each step, the algorithm selects the item which maximizes an objective function f_{obj} (line 1), which in turn can be defined to find a trade-off between accuracy and diversity, and add it to the re-ranked list (line 1). Thus, it requires $\mathcal{O}(N^2 n)$ computations of the function f_{obj}.

Data: The original list $\overline{\mathbf{R}}$, $N \leq n$
Result: The re-ranked list $\overline{\mathbf{S}}$

1 $\overline{\mathbf{S}} = \langle\rangle$;
2 **while** $|\,\mathbf{S}\,| < N$ **do**
3 $\quad i^* = \underset{i \in \mathbf{R}}{\operatorname{argmax}}\ f_{obj}(i, \overline{\mathbf{S}}, u)$;
4 $\quad \overline{\mathbf{S}} = \overline{\mathbf{S}} \circ i^*$;
5 $\quad \mathbf{R} = \mathbf{R} \setminus \{i^*\}$
6 **end**
7 **return** $\overline{\mathbf{S}}$.

Algorithm 1: The greedy strategy. We remind that the overlined capitalized letters are used for lists and capitalized letters for the corresponding sets. The set cardinality is denoted with $|\cdot|$, the $\setminus$ symbol corresponds to set difference and the symbol $\circ$ is used for appending new elements to a list. $\langle\rangle$ indicates an empty list.

As for search results diversification, the diversity in a list of recommendations may be increased in an **implicit** or **explicit** manner [16]. The implicit diversification aims to increase the average distance between pairs of items in the recommendation list, while the explicit one tries to diversify the list by covering the user interests represented via categories or other information that can describe the items. Explicit diversification is also known as Intent-Aware. In fact, user intents in information retrieval correspond to

user interests in recommender systems. Among state-of-the-art diversification algorithms, Maximal Marginal Relevance (MMR) [158] is an implicit approach, while eXplicit QUery Aspect Diversification (xQuAD) [159] represents an explicit strategy.

2.9.2 Maximal Marginal Relevance

MMR **implicitly diversifies** a list considering a trade-off between the relevance of an item and its amount of new information provided with respect to previously selected items. More formally, the objective function of MMR is defined as:

$$f_{obj}(i, \overline{\mathbf{S}}, u) = \lambda \cdot r^*(u, i) + (1 - \lambda) \cdot \operatorname*{avg}_{j \in \mathbf{S}} (1 - sim(i, j)) \tag{2.18}$$

where r^* is a function for rating estimation, sim is a similarity measure on item pairs and the λ parameter lets to manage the accuracy-diversity balance.

2.9.3 Explicit Query Aspect Diversification

Differently from MMR, xQuAD **is an explicit method** since it maximizes the coverage of the inferred interests while minimizing their redundancy. It was proposed for search diversification in information retrieval by Santos et al. [135], as a probabilistic framework to explicitly model an ambiguous query as a set of sub-queries that are supposed to cover the potential aspects of the initial query. More recently, it has been adapted for recommendation diversification by Vargas and Castells [159], replacing query and relative aspects with user and items features, respectively. The expression of the xQuAD objective function is

$$f_{obj}(i, \overline{\mathbf{S}}, u) = \lambda \cdot r^*(u, i) + (1 - \lambda) \cdot div(i, \overline{\mathbf{S}}, u) \tag{2.19}$$

with $div(i, \overline{\mathbf{S}}, u)$ defined as

$$div(i, \overline{\mathbf{S}}, u) = \sum_{f} p(i|f) \cdot p(f|u) \cdot \prod_{j \in \mathbf{S}} (1 - p(j|f)) \tag{2.20}$$

In (2.20) $p(i|f)$ represents the likelihood of item i being chosen given the feature f and is computed as a binary function that returns 1 if the item contains f, 0 otherwise; $p(f|u)$ represents the interest of user u in the feature f and is computed as the relative frequency of the feature f on the items rated by user u. In other words, `xQuAD` fosters the idea of promoting items that are simultaneously highly related to at least one of the features of interest for the user and slightly related to the features of the items already recommended.

2.9.4 Diversity Evaluation Metrics

For evaluating the recommendation quality considering a wide range of evaluation metrics, a number of metrics have been proposed. In this section we show the most known state-of-the-art metrics for evaluation *Individual Diversity*, *Aggregate Diversity*, and *Novelty* in top-N recommendation task. While metrics for accuracy have been already described in Section 2.8.1. Unless explicitly stated, each of the following metrics is computed with respect to a single user and then averaged across all users.

Individual Diversity

The individual diversity of a recommendations set $\mathbf{R}$, whose size will be denoted as $|\mathbf{R}|$ and will match N in a top-N scenario, can be computed as the average dissimilarity of all pairs of items [69]:

$$\mathcal{ILD}(\mathbf{R}) = \frac{1}{|\mathbf{R}| \cdot (|\mathbf{R}| - 1)} \sum_{i \in \mathbf{R}} \sum_{j \in \mathbf{R}, j \neq i} div(i, j) \tag{2.21}$$

The distance function may correspond to the complement of some similarity measure in terms of the item features (content-based view) or their user interaction patterns (collaborative view) [158]. In the content-based version of ILD, when many attributes are considered, we compute $div(i, j)$ as the complement of $sim(i, j) = \text{avg}_{A \in \mathcal{A}}\, sim_A$, where the similarity related to attribute A, sim_A, is given by Jaccard index computation.

Another diversity measure of a recommendation list is Subtopic Recall (S-Recall), proposed for evaluating subtopic retrieval in the information retrieval field, where documents may cover different subtopics of a query topic [174]. Adapted to recommendation task, S-Recall can evaluate the fraction of features covered in a recommendation list. More formally:

$$S{-}Recall(\mathbf{R}) = \text{avg}_{A \in \mathcal{A}} \frac{\left| \bigcup_{i=1}^{N} F_A(i) \right|}{|dom(A)|} \tag{2.22}$$

where $F_A(i)$ represents the set of features of attribute A in the i-th item in $\overline{\mathbf{R}}$. Intuitively, indicating the degree of subtopic coverage, S-Recall also represents the diversity of recommendation list. α-nDCG is the redundancy-aware variant of Normalized Discounted Cumulative Gain proposed in [35]. We adopt the adapted version for recommendation proposed in [146]:

$$\alpha{-}nDCG(\mathbf{R}, u) = \text{avg}_{A \in \mathcal{A}} \frac{1}{\alpha\text{-iDCG}} \cdot \sum_{i=1}^{|\mathbf{R}|} \frac{\sum_{f \in F_A(i)} (1-\alpha)^{cov(\mathbf{R}, f, i-1)}}{log_2(1+i)} \tag{2.23}$$

where $cov(\mathbf{R}, f, i-1)$ is the number of items ranked up to position $i-1$ containing the feature f. The α parameter is used to balance the emphasis between relevance and diversity. α-iDCG denotes the value of α-nDCG for the best "ideally" diversified list. Considering that the computation of the ideal value is NP-complete [35], we adopt a greedy approach: at each step we select solely the item with the highest value, regardless of the next steps.

Aggregate Diversity

Aggregate Diversity represents an important quality dimension for both business and user perspective, since improving the coverage of the items catalog and of the distribution of the items across the users may increase both the sales and the user satisfaction [161]. To evaluate Aggregate Diversity, catalog coverage [57] (the percentage of items in the catalog recommended at least once), and Gini coefficient [7, 161] (for the distribution of recommended items) can be used. The latter is useful to analyse the concentration degree

of top-N recommendations across all items and its scale is reversed, thereby forcing small values to represent low distributional equity and large values to represent higher equity.

$$coverage = \frac{|\bigcup_{u \in U} top\text{-}N(u)|}{|I|} \tag{2.24}$$

$$Gini\ coefficient = 2 \cdot \sum_{i \in I} \left(\frac{|I| + 1 - rank(i)}{|I| + 1} \right) \cdot \left(\frac{rec(i)}{|U|} \right) \tag{2.25}$$

In Equation (2.25) $rec(i)$ is the number of users to whom i has been recommended and $|U|$ is the number of users, while $rank(i)$ is the position of i if items were ordered according to the number of users they have been recommended to. The coverage metric needs to be considered together with a distribution metric like Gini coefficient, since the coverage gives an indication about the ability of a recommender to cover the items catalog, and the other one shows the ability to equally spread out the recommendations across all the items. Hence, only an improvement of both metrics indicates a real increasing of aggregate diversity, that in turn denotes a better personalization of recommendations [7].

Novelty

In some of the works described in this thesis, we evaluate the popularity-based novelty [158] which measures the unexpectedness of an object relative to its global popularity [177]. We use two popularity-based novelty metrics: Expected Popularity Complement (EPC) and the percentage of long-tail items among the recommendations across all users [7] (indicated with *total* in (2.27)) considering the 80 percent of less rated items in the training set as *Long-tail* items.

$$EPC = \frac{\sum_{i \in \mathbf{R}} (1 - pop(i))}{|\mathbf{R}|} \tag{2.26}$$

$$\%Long\text{-}tail = \frac{\sum_{i \in Long\text{-}tail} rec(i)}{total} \tag{2.27}$$

With $pop(i)$ in (2.26) we mean the number of users who rated item i, normalized by the maximum value over the items in the dataset.

The alternative to the popularity-based version is a similarity-based novelty defined upon a distance function between the item and a set of items a user has already interacted with. If the items in her profile are considered, the novelty is called Profile Distance (PD) [158], while the novelty between the current and the previously recommended items is called Temporal Diversity [84]. The similarity-based novelty could use the distance function types already mentioned.

$$PD = \frac{1}{|R| \cdot |train(u)|} \sum_{i \in R} \sum_{j \in train(u)} div(i,j) \tag{2.28}$$

$$\tag{2.29}$$

PD formula is basically similar to ILD one, except that the former compares recommendations and items in the user profile and the latter considers all the possible couples of recommended items.

2.10 Cold Start Problem

A user cold-start situation occurs when a RS does not have enough information about the interests or the past behaviour of a user to provide her good suggestions. Respectively, the item cold-start problem arises when there is little or even no information about historical users' interactions with an item. Typically, users and items are in the cold-start phase when they are new in the system. Cold-start problem is of a great importance in collaborative filtering methods [139, 51], since they strongly rely on the historical information. While content-based RSs can infer the interests of a user even with very few ratings, and can recommend new items since they are not affected by the popularity of the items. Therefore, one of the solutions proposed so far to tackle such problem for both users and items is the use of hybrid RSs, that can take advantages of both content and collaborative information [153].

User Cold Start Problem

Using cross-domain information has been demonstrated to be strongly effective for cold-start users [26, 47], specially for domains strongly related like books and movies. Obviously, it requires information at least about the target users' interests in the source domain, namely the domain used for generating recommendation in the target domain. Another solution is the use of personality information, based on the assumption that users with similar personality traits have similar interests. Such information could be explicitly required to the users or inferred from their behaviour [49]. If cross-domain or personality information is not usable, preferences can be elicited by means of active learning methods, namely requiring directly cold-start users to provide some rating [49]. Finally, another solution, proposed in [142], regards the use of users social network content - e.g. Facebook friends and page likes.

A rigorous cross-validation strategy for evaluation in cold-start scenario is proposed in [78]. It splits each user profile in the test fold into three subsets: training set (N ratings), validation set (M ratings), and testing (remaining ratings, hence at least 1). In order to simulate different user profile sizes from 0 to N likes, training and evaluation are repeated N times, starting with the zero rating in the training set and incrementally increasing it one by one. This particularly methods allows to evaluate each profile size with the same test set, avoiding potential biases in the evaluation due to different test set sizes, as demonstrated in [78]. Zero rating in the training set represents the worst situation, when no information is available for the target user. An example of its application is showed in the Experimental Setting Section 6.4.

Item Cold Start Problem

As said before, for properly recommending new or still unpopular items, an effective solution is to combine content-based and collaborative information in an hybrid algorithm. Another viable solution is to ask users for help, particularly useful when hybrid methods are not applicable for lack of content based information. However, intelligent methods are needed to choose the

right users. The method ExcUseMe, proposed in [12], builds upon a smart exploration algorithm that selects a predefined number of users for exploring new items.

Chapter 3

Adaptive Multi-attribute Diversity

In this chapter, we present an adaptive multi-attribute diversification approach to personalize the recommendation diversity considering the individual inclination of the user to diversifying over different content-based item dimensions. We focus on modeling user propensity toward selecting diverse items, where diversity is computed by means of content-based item attributes. We then exploit such modeling to present a novel approach to re-arrange the list of Top-N items predicted by a recommendation algorithm, with the aim of fostering diversity in the final ranking. An extensive experimental evaluation proves the effectiveness of the proposed approach as well as its ability to improve also novelty and catalog coverage values.

3.1 Introduction

The main task of a recommendation engine is suggesting unknown items in a personalized way and recommend the top N items by considering the highest predicted ratings. As a result, in the recommender systems field new algorithms and approaches have been proposed over the years mostly devoted to maximizing recommendation accuracy. More recently, it has been recognized that improving only the predictive accuracy of recommendations is not enough to judge the effectiveness of a recommender system [23, 98] and several works have tackled the issue of diversification of recommendations as a way to increase user's utility [23, 16, 168, 156, 21], reaching the conclusion that a degree of diversity in the list can be increased at a cost of reducing system accuracy [29]. It is noteworthy that the relation between accuracy and diversity goes beyond a mere trade-off as recently pointed out in [46], where the authors provide an analysis of user's perception of differences in recommendation algorithms and show that there is a strong correlation between perceived accuracy and satisfaction of the users for algorithms able to better diversify the returned list of recommended items. Since diversity is usually characterized as the dissimilarity degree between all the items in the recommendation list [94, 178, 175], one of the most important problems to address is the item-to-item dissimilarity computation. To date, diversity based on only one attribute (e.g. genre in movie and music domains, product category in e-commerce) [156] or collaborative filtering information (e.g. number of co-rating between items) [168] has been mainly considered in the literature.

In some domains, the items description can be enriched by exploiting side information thus moving the items in a multi-dimensional space where each dimension corresponds to a different attribute (e.g. genre, year, director, actor and so on in the movie domain). Indeed, multi-attribute content search and filtering techniques have been proposed to help users in better specifying her preferences or needs, or eliciting them, based on various attributes [8]. However, multi-attribute diversity is still under-explored. The main research

(a)

(b)

(c)

Figure 3.1: Example of three recommendation lists with different degree of diversity: (a) low diversity, all the movies have same actor (Tom Cruise) and genre (Action); (b) the actor is still the same but there are different genres; (c) higher diversity, in terms of both actor and genre. We see how the portfolio effect is more evident in (a) and (b).

questions we address in this chapter are:

- **RQ1** ***How can we model different users' attitude with refer-***

ence to diverse items in the recommendation list?

- **RQ2** ***Does each user need diversity for every attribute?***
- **RQ3** ***What is the right level of diversity for each attribute?***

The main intuitions behind our work are that: (i) users could be inclined to diversifying only with respect to some specific item dimensions (e.g., item attributes as *director* and *year* in the movie domain) and not be interested in diverse suggestions related to other ones (e.g. *genre* in the movie domain); (ii) we can extract this information from the user's past interaction with the system. By way of example, a user can be interested in a movie director more than in its genre. Analogously, a user can strongly prefer a particular actor and accept to watch movies of several genres. As a consequence we need to measure such differences, that is to show the different tendencies of each user in diversifying her choices, dealing with significant item attributes. Following these ideas, we propose an *adaptive multi-attribute* diversification approach able to customize the degree of individual diversity by taking into account the inclination of the user to diversifying over different content-based item dimensions. Specifically, we employ Entropy as a measure for the diversity degree while modeling user preferences and use it in conjunction with the user profile dimension for calibrating the degree of diversification of the list.

This work considerably extends our previous work [43] where the notion of user quadrants defined in terms of attribute-based Entropy and profile dimension was originally introduced to foster the computation of diversified recommendation lists. The new contributions presented in this chapter refer to different aspects of the overall approach. We introduce a new modeling of the user propensity towards diversity which is not based on an exclusive classification in four quadrants but allows the user to belong to all the quadrants to a certain degree (this is the main reason why we call this new modeling *fuzzy approach*). In fact, the classification of users in four quadrants originally proposed in [43] seemed a too strong hypothesis to be of practical use. We also compared how the two different modelings affect recommen-

dation results in terms not just of diversity, but also in terms of accuracy and novelty of recommendation as well as in terms of catalog coverage (a.k.a. aggregate diversity) [7]. We show that our approach to diversification on the one hand reduces the *portfolio effect* while remaining, on the other hand, effective compared to the other evaluation dimensions just mentioned. The two modelings have been tested against two recommendation datasets that refer to different domains. More specifically, the main contributions of this chapter are:

- *Analysis of user needs in terms of individual diversity.* Other than the clustering of users in four disjoint quadrants originally introduced in [43], here we propose a more fine-grained analysis of users profiles introducing a fuzzy classification. For each attribute describing an item and according to the individual values of entropy and profile length, each user belongs to each quadrant with a certain degree.

- *Evaluation Methodology.* We propose an evaluation of our approach for individual diversity by considering also its performance in terms of accuracy, novelty and aggregate diversity. For the evaluation we tested both an implicit (`MMR` [28]) and an explicit (`xQuAD` [135]) method (see Section 2.9.1). The evaluation has been performed by considering Pareto optimal solutions.

- *Empirical Analysis.* We demonstrate the validity of our intuition via an extensive experimental evaluation on two datasets involving several baseline systems.

3.2 Related work

In offline evaluation settings, accuracy and diversity act in opposition with each other, since improving one of them usually leads to shrink the other, as already explained in Section 2.9. The concept of Pareto optimality could be used to face the trade-off of multi-objective problems [114, 126].

Recently, the idea of considering the user interests in the diversification approach in order to personalize the recommendation diversity received increasing interest. User modeling techniques have tried to characterize deeply the users-items interactions and to move beyond the network of users just based on the rating history, as in [96], where a trustworthy network made of users in which a user can rely on has been built. In [159] the identification of diversity within the user profile is carried out through the extraction of user sub-profiles to reflect the polyfacetic nature of user interests, where the definition of a sub-profile is done by analysing only the genre of a movie. The authors of [34] point out a causal relationship between personality factors (such as openness and conscientiousness) and the degree of diversification in the user choices with respect to genres, actors, directors, country or year of release of a movie. As a further validation, in [167], the same authors suggest a solution taking into account personality for generating more personalized diverse recommendations and consolidating their previous observations. [43] proposes one of the first attribute-based diversification approach, which is able to customize the degree of diversity of the recommendation list by taking into account the diversity inclination of the user across different item attributes.

Please refer to the Section 2.9 for a more detailed description of the diversity problem.

3.3 Adaptive multi-attribute diversification

In this section we introduce and describe our proposal to model user attitude towards diversification in recommender systems. Figure 2 shows a possible representation of a recommendation engine that exploits our approach to mitigate the portfolio effect. To this aim, we adopt a re-ranking procedure [7] in an adaptive multi-attribute setting that acts on the recommendations lists provided by a generic recommendation algorithm. Re-ranking has been shown [7] to be effective in increasing diversity in results while not affect-

ing the computational complexity of the overall recommendation procedure. Instead of relying on a multi-objective optimization function that tries to maximize both diversity and accuracy, the recommendation algorithm only takes care of accuracy and leaves to a simpler re-ranking procedure the task of increasing the diversity in the final recommendation list.

Before moving into a detailed description of the diversification procedure we briefly describe the different phases of the complete recommendation scenario.

- *Inputs.* The inputs of the system are: (i) the User-Item matrix where we have the rating history of each user; (ii) a structured description of the items belonging to the catalog. Such information can be extracted from external knowledge sources such as *Wikipedia*, *Google*, *last.fm*, *IMDb*, *MusicBrainz*, etc..

- *User modeling.* Based on the inputs, for each user the system computes a model of her propensity towards diversified recommendation. In our case, for each attribute describing the item, the system evaluates the quadrant the user belongs to (see Section 3.3.1 for more details).

- *Computation of the recommendation list.* The recommendation algorithm exploits the User-Item matrix and optionally the description of the items in the catalog to compute a list of recommended items. If we are interested in returning the *top-N* best items to the user, in this phase the recommendation engine computes the *top-M* best items, with $M > N$. It is noteworthy that we are not interested here in the specific recommendation algorithm as we only focus on the eventual re-ranking phase. Indeed, in our experimental setting (see Section 3.5) we evaluated our diversification model against different state of the art algorithms (`BPRSLIM`, `BPRMF`, `WRMF`, `SoftMarginRankingMF`, `ItemKNN`).

- *Re-ranking.* Based on the user classification into quadrants, the system re-ranks the recommendation list previously computed.

- *Output.* The user is returned with the *top-N* items from the re-ranked list.

Note that this method needs a sufficient quantity of ratings for each user, since it relies on Entropy and profile length information. Therefore, it is not able to work properly for cold-start users, namely those users who have provided an exiguous number of ratings (usually less than 5) or even no rating at all. In such situations, additional information is required. For instance, personality information have proved to be a good solution for facing the cold-start problem [49] and for adjusting diversity [167], although it is not always available or inferable from rating data.

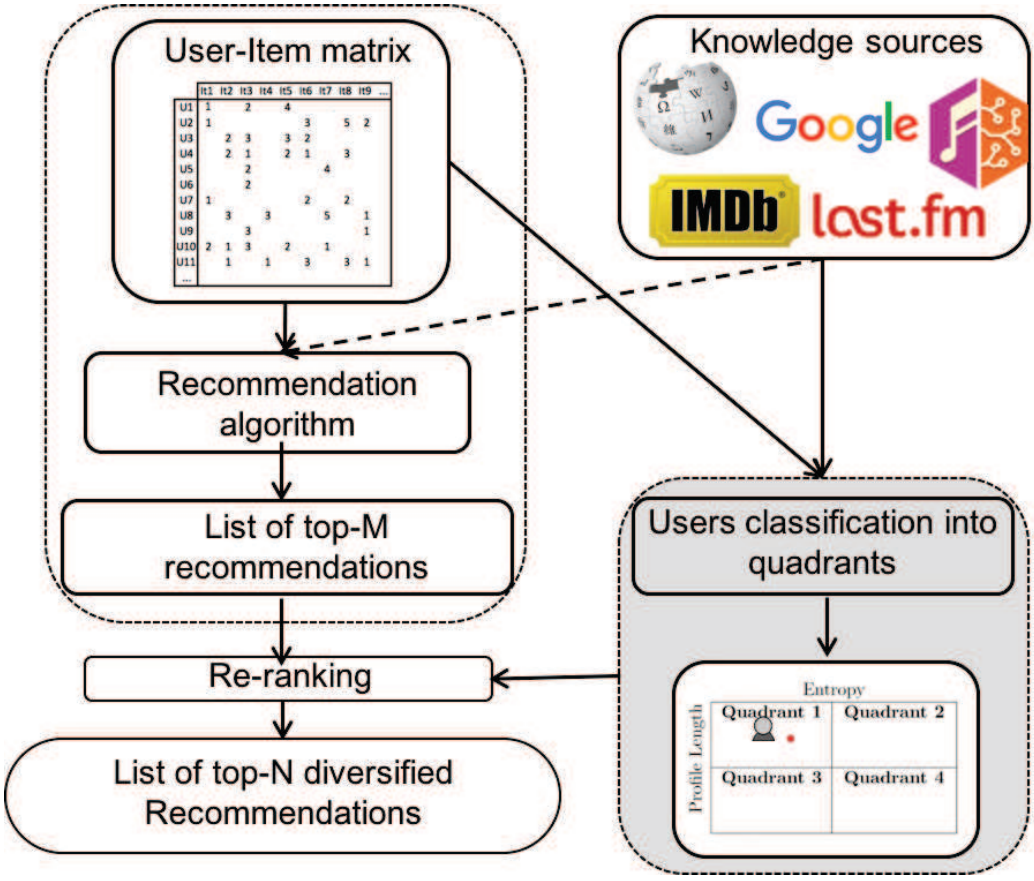


Figure 3.2: A schematic representation of the overall architecture.

In the following we detail how the *adaptive multi-attribute* diversification

approach works. We start by introducing the notion of User Quadrants and then we move to their Fuzzy version. Subsequently, we show how the diversification approaches `MMR` and `xQuAD`, introduced in Section 2.9.3, may be adjusted to adaptive strategies under a multi-attribute setting. In other words, our intent is to modify the objective functions of `MMR` and `xQuAD` such that the diversification attitude of each user with respect to different item attributes (i.e., *year*, *genre*, *director* and *actor* in the movie domain and *genre*, *author* and *subject* in the book domain) could stand out.

3.3.1 User Quadrants

In order to measure user's propensity to diversity on a specific attribute we used Shannon's Entropy which can be used as a measure of the information content associated with an attribute $A \in \mathcal{A}$ for each user u [97]. We compute Entropy with reference to each attribute $A \in \mathcal{A}$ to evaluate the degree of diversity with respect to u. Shannon's Entropy for user u and attribute A with $|dom(A)|$ values can be computed as:

$$\mathcal{H}_A(u) = -\sum_{k=1}^{|dom(A)|} p_k \cdot \log p_k \tag{3.1}$$

where p_k is the relative frequency of the k-th value of A considering all the items (elements) belonging to the user profile (collection of the items rated by the user).

Our model is adaptive in the way that it is based on the classification of users in four groups, referred to as *quadrants*, defined by considering as discriminating parameters the medians of the Entropy distribution and user profile length distribution across all users. A separate clustering is computed for each attribute describing the item. For example a user u is in the first quadrant for the *genre* attribute, if her Entropy $\mathcal{H}_{genre}(u)$ is less than the median of the Entropy computed across all users and she has a short user profile (her number of ratings is less than the median of users' ratings). The same user may belong to different quadrants in relation to different attributes. All the

quadrants are represented in Figure 3.3.

The rationale behind our clustering hypothesis is that we can look at the previous interactions of the user with the system to infer whether she likes to enjoy items which result different with regards to some specific characteristics or not. If she uses to read books of the same subjects regardless of the author we may interpret this behavior as a clue that she is more willing to diversify with reference to authors while she is less willing with reference to subjects. It is noteworthy that such observation is more valid in the presence of longer interaction of the user with the system. We may imagine to have more information from a user who, during her whole interaction with the system, read dozens of books of the same genre from a high variety of authors rather than from a user who read only, say, five books of the same genre from five different authors. In the former case we have a stronger hint about the user attitude towards author diversification than in the latter one. Analogously we may say that the former user has a very low propensity towards genre diversification.

Given an attribute A, a high value of Entropy is then interpreted as an attitude of the user to choose items with different values for A. Conversely, a low value of Entropy is interpreted as her willingness to consider items similar with reference to that attribute. Furthermore, we are considering the user' profile length since we want to allow various values of Entropy to play a different role for users with a large or respectively short interaction with the system, making the Entropy computation potentially more meaningful if supported by a longer user experience.
The quadrants the user belongs to, potentially different for each item attribute, are used to rewrite $sim(i, j)$ in Equation (2.18) and $div(i, \overline{\mathbf{S}}, u)$ in Equation (2.19), as better explained in Sections 3.3.3 and 3.3.4 respectively. Given a user u and the set of item attributes $\mathcal{A}$, we then consider a function $q_u : \mathcal{A} \rightarrow \{1, 2, 3, 4\}$, which assigns, for each attribute, the quadrant to

which user u belongs. Moreover, we introduce an absolute quadrant weight $\omega_k \in [0, 1]$, with $k \in \{1, 2, 3, 4\}$. Of course, more groups can be defined thus identifying more than four quadrants. However, we have already shown in [43] that even with such a coarse grained classification we are able to obtain interesting results in terms of *precision* and *intra list diversity* (ILD) values, and we will see how experiments described in Section 3.5 confirm this trend.

Profile Length \ Entropy		
	Quadrant 1	**Quadrant 2**
	Low Entropy	High Entropy
	Small Profile	Small Profile
	Quadrant 3	**Quadrant 4**
	Low Entropy	High Entropy
	Large Profile	Large Profile

Figure 3.3: Quadrants

3.3.2 Fuzzy Quadrants

Users hard clustering proposed in Section 3.3.1 and tested in our previous work [43] could seem too rigid because of a sharp discrimination into four quadrants. One way to overcome this inconvenience is to introduce a fuzzy users clustering (a.k.a. soft clustering) that permits a user to belong to more than one quadrant simultaneously with a different degree. In fact, we defined functions able to compute a membership degree for each quadrant. This setting can be regarded as the opposite extreme to the hard clustering in just four quadrants of the previous section, as it represents potentially infinite clusters to which a user may belong to. This allows us to get a comparison between the simplest version of clustering by median values and the fine-grained version represented by fuzzy clustering. In order to evaluate the membership grades to quadrants, we reproduced the

quadrants subdivision in the unit square, normalizing in [0,1] the values of Entropy and profile length and considering them as respectively the x and y coordinates. The x and y values then become the inputs for four bivariate Gaussian functions f_1, f_2, f_3, f_4 where $f_1 \sim \mathcal{N}((0,0), \sigma^2)$ (shown in Figure 3.4), $f_2 \sim \mathcal{N}((1,0), \sigma^2)$, $f_3 \sim \mathcal{N}((0,1), \sigma^2)$ and $f_4 \sim \mathcal{N}((1,1), \sigma^2)$. Whenever in the experiments we mention the fuzzy approach, we mean that we substituted the weights $\omega_{q_u(A)}$ introduced in Section 3.3.1 with a weighted sum

$$\omega_{q_u(A)} = \sum_{k=1}^{4} \omega_k \cdot f_k(x, y) \tag{3.2}$$

where x is the value of Entropy and y profile length for user u and ω_k are the absolute quadrants weights for attribute A. Among different membership functions, Gaussian functions are quite popular in the fuzzy logic literature. In particular they provide the advantage that their output is very smooth and have been proved to be most adequate for fuzzy logic [62]. The value of σ^2 is the same for the four functions and is chosen so that for point $(\frac{1}{2}, \frac{1}{2})$ each function assumes the maximum value of f divided by 4 ($\sigma^2 = 0.1803$).

3.3.3 Adaptive MMR

Here we are going to explain how the diversification algorithm `MMR`, introduced in Section 2.9.2, can be adjusted to incorporate the weights computed with User Quadrants or Fuzzy Quadrants settings. As we deal with a multi-attribute problem, sim has to consider similarities with respect to a set of attributes $\mathcal{A}$ and, for each attribute $A \in \mathcal{A}$, $sim_A(i, j)$ will hereafter denote the similarity between item i and item j with relation to A. The overall similarity between item i and item j in Equation (2.18), for the generic user u, becomes tailored to the quadrants she belongs to and is defined as:

$$sim(i, j) = \frac{\sum_{A \in \mathcal{A}} \omega_{q_u(A)} \cdot sim_A(i, j)}{m \cdot |\mathcal{A}|} \tag{3.3}$$

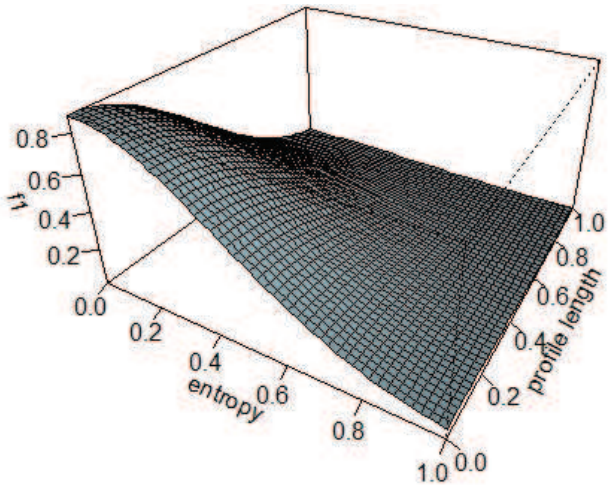


Figure 3.4: Plot of the f_1 function, where $f_1 \sim \mathcal{N}((0,0), \sigma^2)$, used for the fuzzy users clustering. In this case, users with entropy and profile length both close to 0 will receive a value close to 1, indicating a strong belonging to the first quadrant.

with $m = \max\{\omega_k \mid k = 1,2,3,4\}$ and $sim_A(i,j)$ being a similarity measure between i and j with respect to attribute A. The weights associated to quadrants the user belongs to influence the similarity score in Equation (3.3) and hence the resulting objective function of Equation (2.18). Specifically, based on our modeling hypothesis, the weights account for the user propensity in diversifying every single attribute. In fact, if a user is in the second or forth quadrant for a fixed attribute, then assigning a sufficiently big value to ω_2 and ω_4 corresponds to keeping a high value for the original similarity score and thus decreasing the overall value of $f_{obj}(i, \overline{\mathbf{S}}, u)$ for the items i most similar to the ones already available in $\overline{\mathbf{S}}$. These are the items we want to reduce in $\overline{\mathbf{S}}$, in order to guarantee a higher diversity value. Conversely, assigning low weights to the first and third quadrant (low values for ω_1 and ω_3) results in a significant lowering of the original similarity score and hence in an increase of the corresponding $f_{obj}(i, \overline{\mathbf{S}}, u)$ values. This corresponds to preferring items similar to the ones in the re-ranked list $\overline{\mathbf{S}}$.

3.3.4 Adaptive xQuAD

For the intent-aware diversification algorithm `xQuAD`, introduced in Section 2.9.3, we use an adaptation that allows to deal with the multi-attribute problem. Let $\mathcal{A}$ be the set of attributes and let us indicate with $A \in \mathcal{A}$ one of these attributes and with $f \in dom(A)$ the possible values or *features* of A. *div* in Equation (2.20) may be reformulated as follows

$$div(i, \overline{\mathbf{S}}, u) = \sum_{A \in \mathcal{A}} \frac{\sum_{f \in dom(A)} p(i|f) \cdot p(f|u) \cdot (1 - \text{avg}_{j \in \mathbf{S}}\, p(j|f))}{\sum_{f \in dom(A)} p(f|u)} \tag{3.4}$$

While `MMR` contains a simple similarity function where we can inject quadrants weights, `xQuAD` uses Equation (3.4) to compute the diversity across all the attributes via an explicit evaluation of the diversity between the features for each attribute. Therefore, we introduce weights in that formula changing the sum into a weighted sum. More formally, we rewrite $div(i, \overline{\mathbf{S}}, u)$ as

$$div(i, \overline{\mathbf{S}}, u) = \sum_{A \in \mathcal{A}} \omega_{q_u(A)} \cdot \frac{\sum_{f \in dom(A)} p(i|f) \cdot p(f|u) \cdot (1 - \text{avg}_{j \in \mathbf{S}}\, p(j|f))}{\sum_{f \in dom(A)} p(f|u)} \tag{3.5}$$

3.4 Experimental setting

3.4.1 Datasets

In order to test the effectiveness of our proposal for adaptive multi-attribute diversification, we carried out experiments on the well known `Movielens 1M`[1] dataset and on the `LibraryThing`[2] dataset.

`MovieLens 1M` dataset contains 1 million ratings from 6,040 users on 3,952 movies. The original dataset contains information about genres and year of release, and was enriched with side information such as actors and directors

[1] Available at http://grouplens.org/datasets/movielens

[2] Available at http://www.librarything.com/services/

extracted from DBpedia[3]. More details about this enriched version of the dataset are available in [112]. Since not all movies have a corresponding resource in DBpedia, the final dataset contains 998,963 ratings from 6,040 users on 3,883 items. We built training and test sets by employing a 60%-40% temporal split for each user. Moreover, we used the `LibraryThing` dataset, which contains more than 2 million ratings from 7,279 users on 37,232 books. As in the dataset there are many duplicated ratings, when a user has rated more than once the same item, we selected her last rating. The unique ratings are 749,401. Also in this case, we enriched the dataset by mapping the books with BaseKB[4], the RDF version of Freebase[5] and then extracting three meaningful attributes: *genre*, *author* and *subject*. The subjects in Freebase represent the topic of the book, for instance Pilot experiment, Education, Culture of Italy, Martin Luther King and so on. The dump of the mapping is available online[6]. The final dataset contains 565,310 ratings from 7,278 users on 27,358 books. We built training and test sets by employing a 80%-20% hold-out split. The different ratio used for `LibraryThing` compared to `Movielens` (60%-40%) depends on its higher sparsity: holding 80% to build the user profile ensures a sufficient number of ratings to train the system.

	Movielens	**LibraryThing**
Number of users	6,040	7,278
Number of items	3,883	27,358
Number of ratings	998,963	565,310
Data sparsity	95.7%	99.7%
Avg users per item	275.57	20.66
Avg items per user	165.39	77.68

Table 3.1: **Statistics about the two datasets**

Since the number of distinct values was too large for *year*, *actors* and *di-*

[3] http://dbpedia.org
[4] http://basekb.com
[5] https://www.freebase.com
[6] http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/BaseKB2LibraryThing.zip

rector attributes in `Movielens` and for all the attributes in LibraryThing, we convert years in the corresponding decades and performed a K-means clustering for other attributes on the basis of DBpedia categories for `Movielens` and Freebase classes for `LibraryThing`. Table 3.2 and 3.3 report the number of attribute values and clusters. The number of clusters was decided according to the calculation of the within-cluster sum of squares (*withiness* measure from the R Stats Package, version 2.15.3), that is picking the value of K corresponding to an evident break in the distribution of the *withiness* measure against the number of extracted clusters.

	Num. Values	**Num. Clusters**
Genres	19	-
Decades	10	-
Actors	14736	20
Directors	3194	20

Table 3.2: **Statistics about `Movielens` attributes**

	Num. Values	**Num. Clusters**
Genres	270	30
Authors	12868	22
Subjects	2911	20

Table 3.3: **Statistics about `LibraryThing` attributes**

3.4.2 Recommendation Algorithms

Differently from [43], where the baseline was a generic user-based kNN Collaborative Filtering algorithm using Pearson correlation as similarity measure, here, for both datasets we adopt five different algorithms as baselines. We selected five state of the art algorithms available in MyMediaLite[7]:

[7] http://www.mymedialite.net

`BPRSLIM`, `BPRMF`, `WRMF`, `SoftMarginRankingMF` and `ItemKNN`. They were used to create a list of 200 recommendations to build the initial list $\overline{\mathbf{R}}$ used for performing the re-ranking step shown in Algorithm 1. With reference to Equation (2.18) and Equation (2.19) they represent $r^*(u, i)$. Jaccard index was used to compute $sim_A(i, j)$, as in [158, 168, 60], because each feature is represented by a binary value for each item: 1 if present, 0 otherwise[8].

3.4.3 Preliminary insight into Movies Recommendation

Before discussing all the final experimental results, it may be worth to remind some preliminary results that motivated the extensive experiments on the here proposed adaptive multi-attribute diversification approach, related to an initial analysis presented in [43]. Such analysis uses, as baseline, a user-based K-Nearest Neighbors Collaborative Filtering algorithm with Pearson correlation as similarity measure (see Section 2.2 and Equation 2.1). The comparison is between (i) recommendations generated considering the ratings predicted by the baseline without diversification, denoted as `no-div`; (ii) recommendations generated as in (i) and then diversified by the algorithm `MMR`, indiscriminately to all users regardless of whether they are incline to diversifying their choices or not. The target of this experiment is to analyse the trade-off between accuracy and diversity, under the perspective of the classification of users into quadrants showed in 3.3.1. The metrics involved in this analysis are $Precision$@10 and ILD@10, and the λ for `MMR` is fixed to 0.5.

This preliminary comparative experiment corroborates the modelling hypothesis behind the approach proposed in this chapter, and explicitly the fact that users who have explored items in the past are more favourable to diversity. Specifically, the performances of the two algorithms `no-div` and `MMR` are compared on different groups of users, considering those belonging

[8]Cosine distance could be used to compute the distance between two items, but it is more appropriate in presence of weighted values [168].

	Quadrant 1 (1149 users)		*Quadrant 2* (469 users)	
algorithm	**P@10**	**ILD@10**	**P@10**	**ILD@10**
UNN	**0.0455**	0.3890	0.0678	0.3663
UNN + MMR	0.0394	**0.4363**	**0.0706**	**0.4212**

	Quadrant 3 (467 users)		*Quadrant 4* (1146 users)	
algorithm	**P@10**	**ILD@10**	**P@10**	**ILD@10**
UNN	**0.0904**	0.3961	0.1306	0.3544
UNN + MMR	0.0829	**0.4355**	**0.1325**	**0.4012**

Table 3.4: Accuracy and Diversity Results distributed among the different quadrants. *Quadrant 1* contains users belonging to Quadrant 1 for at least 3 attributes; analogously for the other quadrants (users who belong for each attribute to quadrant 2 are only 69, while those belonging always to quadrant 3 are 70).

to the same quadrant for at least three attributes. The results in Table 3.4 show that `MMR` overcomes `no-div` for quadrant 2 and 4 for both Precision and ILD, demonstrating that users with high entropy benefit from diversification. In the other quadrants (1 and 3) there is instead a normal decrease of accuracy when ILD grows. This confirms the intuition that users with low diversity in the user profile are not inclined to an uncontrolled diversification. Therefore, this preliminary analysis supports the adoption of entropy as discriminant parameter for users' classification and motivates further the interest into a recommendation approach that is personalized with respect to diversification. Another insight arising from this analysis regards the possibility to foster the recommendation diversity without affecting accuracy or even slightly improve it when a more targeted diversification approach is applied.

3.5 Experimental Results

We conducted a comparative analysis of the adaptive methods we propose, the baselines without diversification introduced in Section 4.4 and the pure

diversification algorithms (MMR and xQuAD). These latter consist of computing recommendations by using respectively Equation (2.18) and Equation (2.19) without considering the adaptive models. This implies that the diversification is applied indiscriminately to all users regardless of whether they are incline to diversifying their choices or not.

In the following we will indicate with $\texttt{MMR}_{quadr}$ the algorithm that carries out a hard users clustering and, given the list returned by the current baseline, performs re-ranking according to (2.18) with (3.3), as explained in Section 3.3.1. The $\texttt{MMR}_{fuzzy}$ model instead consists of a fuzzy clustering of users in four quadrants, as introduced in Section 3.3.2 and with quadrant weights as in Equation (3.2). Analogously, $\texttt{xQuAD}_{quadr}$ and $\texttt{xQuAD}_{fuzzy}$ represent the corresponding configurations for the diversification algorithm xQuAD.

It is common knowledge that building multi-objective recommender systems that suggest items that are simultaneously accurate and diversified may lead to a conflicting-objective problem, where the attempt to improve an objective further may result in worsening other competing objectives. We face the trade-off of multi-objective problems using the concept of Pareto optimality [126], according to which an individual (meant as the result of an algorithm in our case) dominates another if it performs better in at least one of the objectives considered. The Pareto Frontier is the set of all non-dominated individuals: none of them can get better without making at least one individual getting worse. We carried out the same type of comparative analysis based on Pareto frontier for MMR and xQuAD. In those analyses we vary the available parameters: only the value of λ can be modified for the diversification baselines, while λ and the quadrant weights w_1, w_2, w_3, w_4 are modified for *quadr* and *fuzzy* algorithms. The step size for variation was fixed in 0.05 for both λ and w_1, w_2, w_3, w_4. The results of this analysis are shown in Figures 3.5, 3.6, 3.7 and 3.8. However, a Pareto Frontier consists in potentially many individuals and in a realistic scenario the system designer would want to choose one or a few of them. In [126], an individual is chosen by means of a linear search on all of the individuals, selecting the one

which maximizes a weighted mean on the objectives in the objective vector, where the weights in the weighted mean represent the priority given to each objective. For instance, if the objectives are accuracy and diversity, the objective vector [Accuracy = 0.7, Diversity = 0.3] allows the system to find the individual that strongly preserves the accuracy and slightly improves the diversity. In this work, in order to demonstrate the validity of the proposed adaptive diversification approach, we carried out a further comparison of the analysed algorithms selecting the most accurate individuals and those with the best mean between accuracy and diversity. Results are shown in Tables 3.5–3.12.

3.5.1 Comparative Results for MMR

The curves in Figures 3.5 and 3.6 show the relation between precision and other metrics, respectively for `Movielens` and `LibraryThing`, using the baseline `BPRSLIM` and the diversification algorithm `MMR`. Focusing on individual diversity, they point out that there is no particular difference in terms of ILD and α-nDCG, but there are improvements considering S-Recall. It means that using the adaptive models there is not an actual direct improvements on individual diversity, but the number of retrieved subtopics increases. Analysing aggregate diversity, the adaptive models improve both coverage and Gini coefficient, which indicates a real increment of aggregate diversity, as explained in Section 2.9.4. In particular, $\texttt{MMR}_{quadr}$ leads to a broader range of values compared to the diversification baseline and $\texttt{MMR}_{fuzzy}$. When considering the novelty dimension, $\texttt{MMR}_{quadr}$ leads to the best results, especially in terms of EPC, namely the popularity complement of the recommended items, while there is no relevant difference between $\texttt{MMR}_{fuzzy}$ and `MMR`. The trends of the results are substantially similar across the other different baselines (`BPRMF`, `WRMF`, `SoftMarginRankingMF`, `ItemKNN`), whose results are shown in the Appendix B.

Beyond the Pareto frontier, which is useful to point out the compromise between the involved objectives through many individuals, we analysed specific

individuals as shown in Tables 3.5, 3.6, 3.7 and 3.8 [126]. We selected the best individuals for the algorithms involved in each comparison, according to two configurations. The first one considers as objective just accuracy, specifically Precision, while the second one corresponds to an unbiased balance between accuracy and diversity (ILD).

As already shown in [43], calibrating the diversity among different content-based attributes may lead to enhance diversity without penalizing accuracy. The same surprisingly good performance is observed in Tables 3.5 and 3.6 with the most accurate individuals for compared algorithms on `Movielens` and `LibraryThing`, respectively. Consistently with the accuracy-diversity trade-off, the basic MMR approach improves the diversity at the cost of the accuracy. The adaptive approaches we propose gain statistically significant improvements with respect to the baseline in terms of diversity, as we expected, but also accuracy, though non statistically significant. Furthermore the adaptive approaches significantly overcome `MMR` in terms of all the metrics, except for S-Recall on `Movielens` where $\texttt{MMR}_{fuzzy}$ obtains the same value of `MMR`. The individuals of Tables 3.7 and 3.8, related to the balance of accuracy and diversity on `Movielens` and `LibraryThing` respectively, improve the diversity with respect to the baseline, closely approaching the values reached by `MMR`, keeping high the accuracy values. A further observation corroborating our modeling hypothesis concerns the weights configurations $\langle \omega_1, \omega_2, \omega_3, \omega_4 \rangle$ for the best individuals found. It is useful to recall that, as explained in Section 3.3, we introduced quadrants weights in the attempt to provide recommendations with a diversity degree reflecting users propensity towards diversification. As a general trend, ω_4 gains the highest values while ω_1 and ω_3 lowest values, which basically means that users with higher entropy and longer profile will receive more diverse recommendations with respect to the other users. Interestingly, ω_2 shows a discordant behaviour between `Movielens` and `LibraryThing` using $\texttt{MMR}_{fuzzy}$. This could be a clue saying that for small profiles the propensity towards diversification is domain dependent and needs more investigations. As we will see in the next section,

the same behaviour holds for xQuAD.

	weights	λ	**Precision**	**Recall**	**nDCG**	**ILD**	**S-Recall**	**α-nDCG**
BS			0.1488	0.0692	0.1634	0.3551	0.2310	0.2773
MMR		0.95	0.1484^{a}	$0.0686^{\mathbf{a}}$	0.1630^{a}	$0.3579^{\mathbf{a}}$	$0.2314^{\mathbf{a}}$	$0.2786^{\mathbf{a}}$
QUADR	⟨0.0, 0.0, 0.2, 0.8⟩	0.55	0.1492^{b}	$0.0690^{\mathbf{b}}$	0.1637	$0.3629^{\mathbf{ab}}$	$0.2321^{\mathbf{ab}}$	$0.2806^{\mathbf{ab}}$
FUZZY	⟨0.0, 0.0, 0.1, 0.9⟩	0.6	$0.1490^{\mathbf{b}}$	$0.0689^{\mathbf{b}}$	$0.1636^{\mathbf{a}}$	$0.3585^{\mathbf{ab}}$	$0.2314^{\mathbf{a}}$	$0.2789^{\mathbf{ab}}$

Table 3.5: Most accurate individuals from Pareto Frontiers for Movielens Dataset, using *BPRSLIM* and MMR. The superscripts a and b indicate statistically significant differences (Wilcoxon signed rank with $p < 0.05$) with respect to the baseline and MMR algorithms, respectively. Bold superscripts indicate stronger statistically significant differences (Wilcoxon signed rank with $p < 0.001$)

	weights	λ	**Precision**	**Recall**	**nDCG**	**ILD**	**S-Recall**	**α-nDCG**
BS			0.0132	0.0146	0.0180	0.3993	0.1375	0.2836
MMR		0.95	0.0131	0.0145	0.0179^{a}	$0.4099^{\mathbf{a}}$	$0.1385^{\mathbf{a}}$	$0.2859^{\mathbf{a}}$
QUADR	⟨0.1, 0.3, 0.0, 0.6⟩	95	0.0135	0.0149	$0.0184^{\mathbf{b}}$	$0.4039^{\mathbf{ab}}$	$0.1375^{\mathbf{ab}}$	$0.2846^{\mathbf{ab}}$
FUZZY	⟨0.0, 0.9, 0.0, 0.1⟩	85	0.0134	0.0147	0.0183^{a}	$0.4100^{\mathbf{ab}}$	$0.1379^{\mathbf{ab}}$	$0.2858^{\mathbf{ab}}$

Table 3.6: Most accurate individuals extracted from Pareto Frontiers for LibraryThing Dataset, using *BPRSLIM* and MMR

	weights	λ	**Precision**	**Recall**	**nDCG**	**ILD**	**S-Recall**	**α-nDCG**
BS			0.1488	0.0692	0.1634	0.3551	0.2310	0.2773
MMR		0.3	$0.1377^{\mathbf{a}}$	$0.0569^{\mathbf{a}}$	$0.1509^{\mathbf{a}}$	$0.4203^{\mathbf{a}}$	$0.2391^{\mathbf{a}}$	$0.2999^{\mathbf{a}}$
QUADR	⟨0.1, 0.1, 0.2, 0.6⟩	0.15	$0.1417^{\mathbf{ab}}$	$0.0616^{\mathbf{ab}}$	$0.1554^{\mathbf{ab}}$	$0.4109^{\mathbf{ab}}$	$0.2377^{\mathbf{ab}}$	$0.2970^{\mathbf{ab}}$
FUZZY	⟨0.0, 0.1, 0.3, 0.6⟩	0.1	$0.1405^{\mathbf{ab}}$	$0.0610^{\mathbf{ab}}$	$0.1541^{\mathbf{ab}}$	$0.4151^{\mathbf{ab}}$	0.2395^{ab}	$0.2986^{\mathbf{ab}}$

Table 3.7: Individuals with best mean between Precision and ILD from Pareto Frontiers for Movielens Dataset, using *BPRSLIM* and MMR

	weights	λ	**Precision**	**Recall**	**nDCG**	**ILD**	**S-Recall**	**α-nDCG**
BS			0.0132	0.0146	0.0180	0.3993	0.1375	0.2836
MMR		0.7	$0.0123^{\mathbf{a}}$	$0.0133^{\mathbf{a}}$	$0.0168^{\mathbf{a}}$	$0.4486^{\mathbf{a}}$	$0.1420^{\mathbf{a}}$	$0.2896^{\mathbf{a}}$
QUADR	⟨0.1, 0.1, 0.3, 0.5⟩	0.4	$0.0123^{\mathbf{a}}$	$0.0134^{\mathbf{a}}$	$0.0168^{\mathbf{ab}}$	$0.4591^{\mathbf{ab}}$	$0.1425^{\mathbf{ab}}$	$0.2898^{\mathbf{a}}$
FUZZY	⟨0.1, 0.6, 0.1, 0.2⟩	0.75	$0.0129^{\mathbf{ab}}$	$0.0140^{\mathbf{ab}}$	$0.0176^{\mathbf{ab}}$	$0.4355^{\mathbf{ab}}$	$0.1407^{\mathbf{ab}}$	$0.2887^{\mathbf{ab}}$

Table 3.8: Individuals with best mean between Precision and ILD extracted from Pareto Frontiers for LibraryThing Dataset, using *BPRSLIM* and MMR

3.5.2 Comparative Results for xQuAD

In this section we investigate the results of the proposed adaptive methods used with the diversification baseline xQuAD. Figures 3.7 and 3.8 show the curves between precision and different other metrics, respectively for Movielens and LibraryThing, using the baseline BPRSLIM and the diversification algorithm xQuAD. Using the Movielens dataset, the adaptive models $\texttt{xQuAD}_{quadr}$ and $\texttt{xQuAD}_{fuzzy}$ lead to improvements in terms of ILD and α-nDCG, and reductions in terms of S-Recall. With regard to aggregate diversity, $\texttt{xQuAD}_{quadr}$ is able to improve coverage and Gini coefficient, while $\texttt{xQuAD}_{fuzzy}$ improves the Gini coefficient but not the coverage reached by xQuAD. Analysing novelty of recommendations, $\texttt{xQuAD}_{quadr}$ gives the highest values of EPC with small loss of Precision and best balance between Precision and EPC. It is noteworthy that the same trend on EPC occurs using $\texttt{MMR}_{quadr}$ on Movielens, as we may see in Figure 3.5.
Considering the LibraryThing dataset, there are relevant differences with MMR. $\texttt{xQuAD}_{quadr}$ and $\texttt{xQuAD}_{fuzzy}$ overcome xQuAD only in terms of S-Recall, while there is no evident difference in terms of ILD. α-nDCG shows a critical situation: $\texttt{xQuAD}_{fuzzy}$ gives the worst results while $\texttt{xQuAD}_{quadr}$ is able to increase the α-nDCG with non significant losses of precision. Moreover, $\texttt{xQuAD}_{quadr}$ and $\texttt{xQuAD}_{fuzzy}$ overcome xQuAD in terms of both coverage and Gini coefficient, therefore giving a real improvement of aggregate diversity. In particular, $\texttt{xQuAD}_{quadr}$ gives the highest values and the best compromise between accuracy and aggregate diversity. Also analyzing novelty of recommendations, $\texttt{xQuAD}_{quadr}$ and $\texttt{xQuAD}_{fuzzy}$ overcome xQuAD, giving better balance between precision and %Long-Tail and between precision and EPC[9].

Just as for MMR, we show in Tables 3.9, 3.10, 3.11 and 3.12 the best individuals of compared algorithms according to the configurations of objectives described above. The situation is analogous to the one depicted for MMR, since

[9] According to the Figures shown in the Appendix B, the aforementioned trends on the results are generally confirmed using the adaptive diversification models upon other recommendation algorithms (BPRMF, WRMF, SoftMarginRankingMF, ItemKNN).

the main outcome is that our adaptive multi-attribute approaches $\texttt{xQuAD}_{quadr}$ and $\texttt{xQuAD}_{fuzzy}$ are able to improve the diversity without accuracy loss, while the pure xQuAD increases the diversity penalizing the accuracy, as expected. The statistically significance test validate the results even further, especially for diversity measures. The same considerations made for MMR on weights ω_4, ω_1 and ω_3 are still effective, and an analogous discordant behaviour for ω_2 can be observed too. In fact, for $\texttt{xQuAD}_{quadr}$ on LibraryThing in both configurations of objectives and $\texttt{xQuAD}_{fuzzy}$ on Movielens just in the first configuration, the value ω_2 is even higher than ω_4, while is almost zero in the other cases.

3.5.3 Results discussion

Summing up, previous results show that our proposed adaptive diversifications model is able to foster the recommendations quality in a multi-objective scenario. More specifically, considering the individual diversity, MMR benefits from the adaptive model in terms of S-Recall on both the datasets, while there is no significant difference in terms of ILD and α-nDCG. It is worth to note that the basic MMR gives the best results in terms ILD and S-Recall among different diversification algorithms, as demonstrated in [156] and here the results show that it is possible to further improve those metrics with the $\texttt{MMR}_{quadr}$ and $\texttt{MMR}_{fuzzy}$. Moreover, it always obtains improvements considering novelty and aggregate diversity, especially using $\texttt{MMR}_{quadr}$. On the other hand, the adaptive models applied with xQuAD show different behaviours on the two datasets, especially considering the individual diversity. They are able to improve the ILD and α-nDCG results on LibraryThing, but not S-Recall, while on Movielens they improve S-Recall and only $\texttt{xQuAD}_{quadr}$ gives better results in terms of α-nDCG. As for MMR, also xQuAD obtains better results in terms of novelty and aggregate diversity, specially using $\texttt{xQuAD}_{quadr}$. In other words, the results suggest that generally using an adaptive model may improve all the balances between accuracy and the other quality dimensions, or at least improve some of them and do not make the other worse.

	weights	λ	**Precision**	**Recall**	**nDCG**	**ILD**	**S-Recall**	**α-nDCG**
BS			0.1488	0.0692	0.1634	0.3551	0.2310	0.2773
XQUAD		0.95	0.1479^{a}	0.0676^{a}	0.1621^{a}	$0.3633^{\mathbf{a}}$	$0.2339^{\mathbf{a}}$	$0.2815^{\mathbf{a}}$
QUADR	0.0,0.2,0.1,0.7	0.8	0.1494^{b}	0.0692	0.1638	$0.3631^{\mathbf{ab}}$	$0.2330^{\mathbf{ab}}$	$0.2806^{\mathbf{ab}}$
FUZZY	0.1,0.5,0.1,0.3	0.95	$0.1489^{\mathbf{b}}$	0.0688^{b}	0.1634^{b}	$0.3575^{\mathbf{ab}}$	$0.2315^{\mathbf{ab}}$	$0.2784^{\mathbf{ab}}$

Table 3.9: Most accurate individuals from Pareto Frontiers for `Movielens` Dataset, using *BPRSLIM* and `xQuAD` The superscripts a and b indicate statistically significant differences (Wilcoxon signed rank with $p < 0.05$) with respect to the baseline and `xQuAD` algorithms, respectively. Bold superscripts indicate stronger statistically significant differences (Wilcoxon signed rank with $p < 0.001$)

	weights	λ	**Precision**	**Recall**	**nDCG**	**ILD**	**S-Recall**	**α-nDCG**
BS			0.0132	0.0146	0.0180	0.3993	0.1375	0.2836
XQUAD		0.95	0.0131	0.0145	0.0179^{a}	$0.4099^{\mathbf{a}}$	$0.1385^{\mathbf{a}}$	$0.2859^{\mathbf{a}}$
QUADR	0.1,0.8,0.0,0.1	0.90	0.0135	0.0152	0.0184	$0.4123^{\mathbf{ab}}$	$0.1404^{\mathbf{ab}}$	$0.2867^{\mathbf{ab}}$
FUZZY	0.3,0.2,0.0,0.5	0.90	0.0134	0.0152	0.0183^{a}	$0.4165^{\mathbf{ab}}$	$0.1410^{\mathbf{ab}}$	$0.2864^{\mathbf{a}b}$

Table 3.10: Most accurate individuals extracted from Pareto Frontiers for `LibraryThing` Dataset, using *BPRSLIM* and `xQuAD`

	weights	λ	**Precision**	**Recall**	**nDCG**	**ILD**	**S-Recall**	**α-nDCG**
BS			0.1488	0.0692	0.1634	0.3551	0.2310	0.2773
XQUAD		0.8	0.1433^{a}	$0.0620^{\mathbf{a}}$	$0.1566^{\mathbf{a}}$	$0.3859^{\mathbf{a}}$	$0.2405^{\mathbf{a}}$	$0.2907^{\mathbf{a}}$
QUADR	0.1,0.1,0.1,0.7,	0.35	$0.1401^{\mathbf{ab}}$	$0.0578^{\mathbf{ab}}$	$0.1528^{\mathbf{ab}}$	$0.4143^{\mathbf{ab}}$	$0.2395^{\mathbf{ab}}$	$0.2966^{\mathbf{ab}}$
FUZZY	0.1,0.2,0.0,0.7,	0.35	$0.1401^{\mathbf{ab}}$	$0.0581^{\mathbf{ab}}$	$0.1528^{\mathbf{ab}}$	$0.4142^{\mathbf{ab}}$	$0.2401^{\mathbf{ab}}$	$0.2968^{\mathbf{ab}}$

Table 3.11: Individuals with best mean between Precision and ILD from Pareto Frontiers for `Movielens` Dataset, using *BPRSLIM* and `xQuAD`

	weights	λ	**Precision**	**Recall**	**nDCG**	**ILD**	**S-Recall**	**α-nDCG**
BS			0.0132	0.0146	0.0180	0.3993	0.1375	0.2836
XQUAD		0.7	$0.0123^{\mathbf{a}}$	$0.0133^{\mathbf{a}}$	$0.0168^{\mathbf{a}}$	$0.4486^{\mathbf{a}}$	$0.1420^{\mathbf{a}}$	$0.2896^{\mathbf{a}}$
QUADR	0.1,0.6,0.1,0.2,	0.9	$0.0134^{\mathbf{b}}$	$0.0151^{\mathbf{b}}$	$0.0183^{a\mathbf{b}}$	$0.4165^{\mathbf{ab}}$	$0.1410^{\mathbf{ab}}$	$0.2864^{\mathbf{ab}}$
FUZZY	0.1,0.1,0.0,0.8,	0.9	$0.0134^{\mathbf{b}}$	$0.0152^{\mathbf{b}}$	$0.0183^{a\mathbf{b}}$	$0.4165^{\mathbf{ab}}$	$0.1410^{\mathbf{ab}}$	$0.2864^{\mathbf{ab}}$

Table 3.12: Individuals with best mean between Precision and ILD extracted from Pareto Frontiers for `LibraryThing` Dataset, using *BPRSLIM* and `xQuAD`

As an additional consideration, the values for quadrant weights proposed in Tables from 3.5 to 3.12 give worth and effectiveness to our idea of using

profile size and entropy to cluster users into groups and approaching their predilection to diversity through belonging groups.

The main difference between the hard clustering and the fuzzy one is that the former assumes that a user can belong to only a quadrant for each attribute, while the second lets a user belong to different quadrants simultaneously with different degree for the same attribute. As a consequence, the hard clustering is straighter, while the fuzzy version tends to distribute more equally the quadrants weights since each quadrants gives a more or less significant contribution. Although the hard version is a simple clustering of users by means of median values, the results point out a positive impact of a clear division of users on most of the evaluation metrics. The hard clustering is able to beat both the fuzzy one and the diversification baseline in terms of Aggregate Diversity (coverage and Gini coefficient) and also in terms of novelty. This is more evident when considering EPC. On the other hand, fuzzy clustering remains very close to the diversification baseline. The reason of this outcome could be found in the aforementioned difference: the hard clustering is more straight than the fuzzy one, therefore it is much more selective during the re-raking phase.

3.6 Summary

Computing effective recommendations calls for approaches which are able to provide not just accurate lists of results. Modern recommendation engines need to go beyond accuracy and consider, while computing a recommendation list, also other dimensions such as diversity in the recommendation list to reduce the *portfolio effect*, catalog coverage to maximize the number of items in the catalog recommended to the users and novelty of results to mitigate the popularity bias thus suggesting also items in the long tail. In particular, it has been shown [46] that reducing the portfolio effect by increasing diversity in the recommendation list plays an important role on user satisfaction. The task is not trivial especially when we deal with a multi-attribute personalized

diversification results. In the recent years, the importance of adapting the recommendation diversity to user's needs with respect to different attributes has strongly emerged, although research on multi-attribute diversity is still in its early stage.

In order to fill this gap, in this work we introduced an adaptive multi-attribute diversification method according to the hypothesis that a user who selected many diverse items in the past could be more willing to receive diverse recommendations. With reference to the research questions pointed out in Section 4.1, as an answer to question RQ1, we proposed to model the user profile by taking into account her attitude to enjoy (or not) items which result diverse with regard to different attributes and eventually adopt this modeling to foster diversity in the list returned by a recommendation engine. Our modeling has been exploited to re-rank the list of items produced by whatever recommender system to reduce the portfolio effect. In order to evaluate the effectiveness of our hypothesis we tested two different versions of our profile modeling (we called them hard and fuzzy), built upon two different state-of-the-art diversification methods - `MMR` and `xQuAD`- in the movie domain on `Movielens 1M` dataset and in the book domain on `LibraryThing` dataset. As for the evaluation of the recommendation quality we considered a wide range of metrics to measure four important quality dimensions: Accuracy, Individual Diversity, Aggregate Diversity, and Novelty in top-N recommendation task. As an answer to question RQ2, the experimental results confirmed our intuition on the need of tailoring diversity degree to actual user's interests in a personalized way, pointing out the inadequate performances of non adaptive diversification baselines. Finally, our approach can be considered as a step forward to solve the challenge posed by question RQ3. In fact, the construction of a content-based user profile in terms of diversity allowed not only to customize the degree of individual diversity in the recommendation list but led to better recommendation quality in a multi-objective scenario. In particular, our adaptive model overcame the traditional accuracy-diversity trade-off issue, improving different qual-

ity objectives, without affecting the others. Hence, the results let us draw the conclusion that diversification methods tailored to actual user's needs produce better recommendations from a broad user utility perspective.

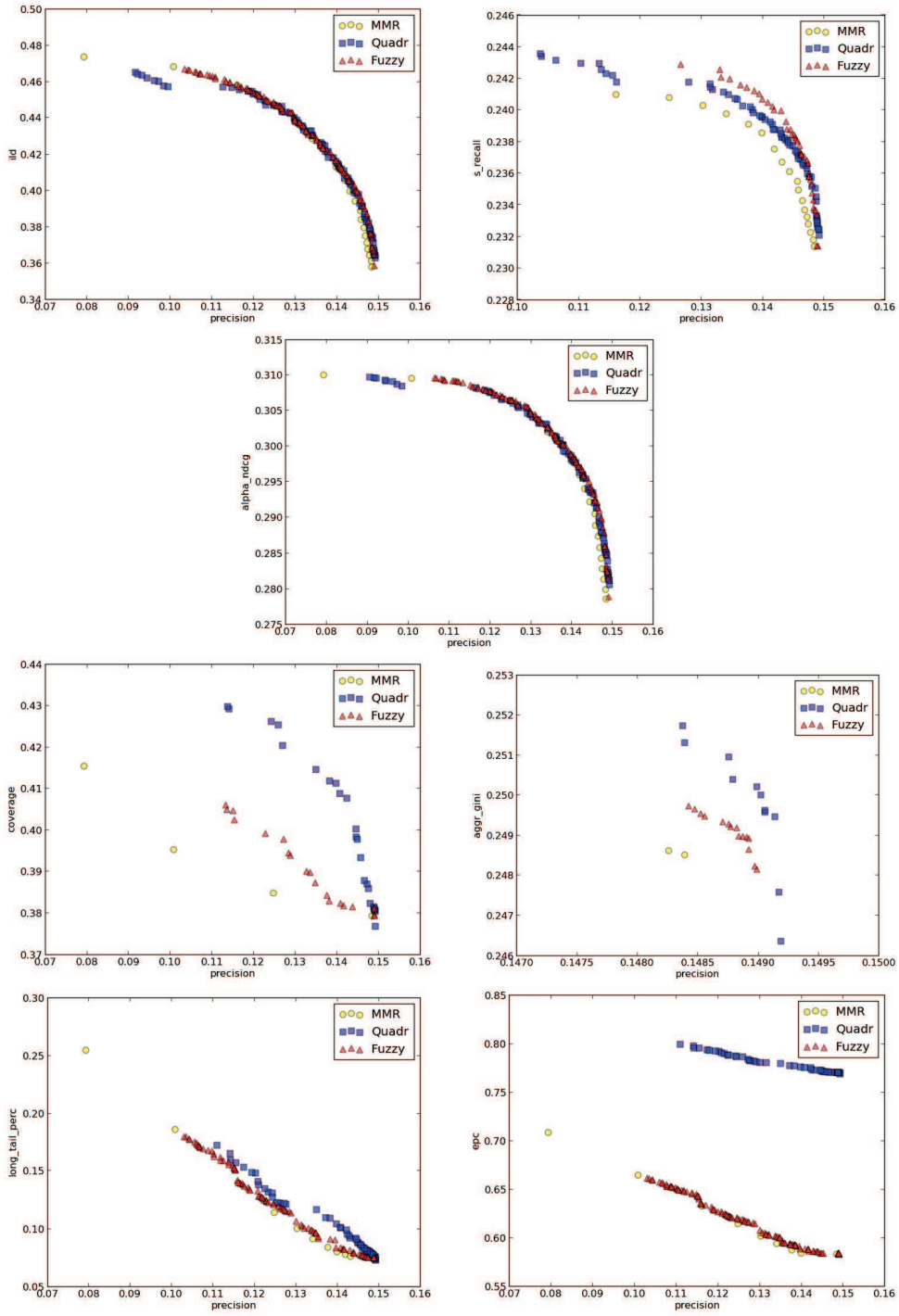


Figure 3.5: Pareto Frontiers for `Movielens` Dataset, using *BPRSLIM* and `MMR`

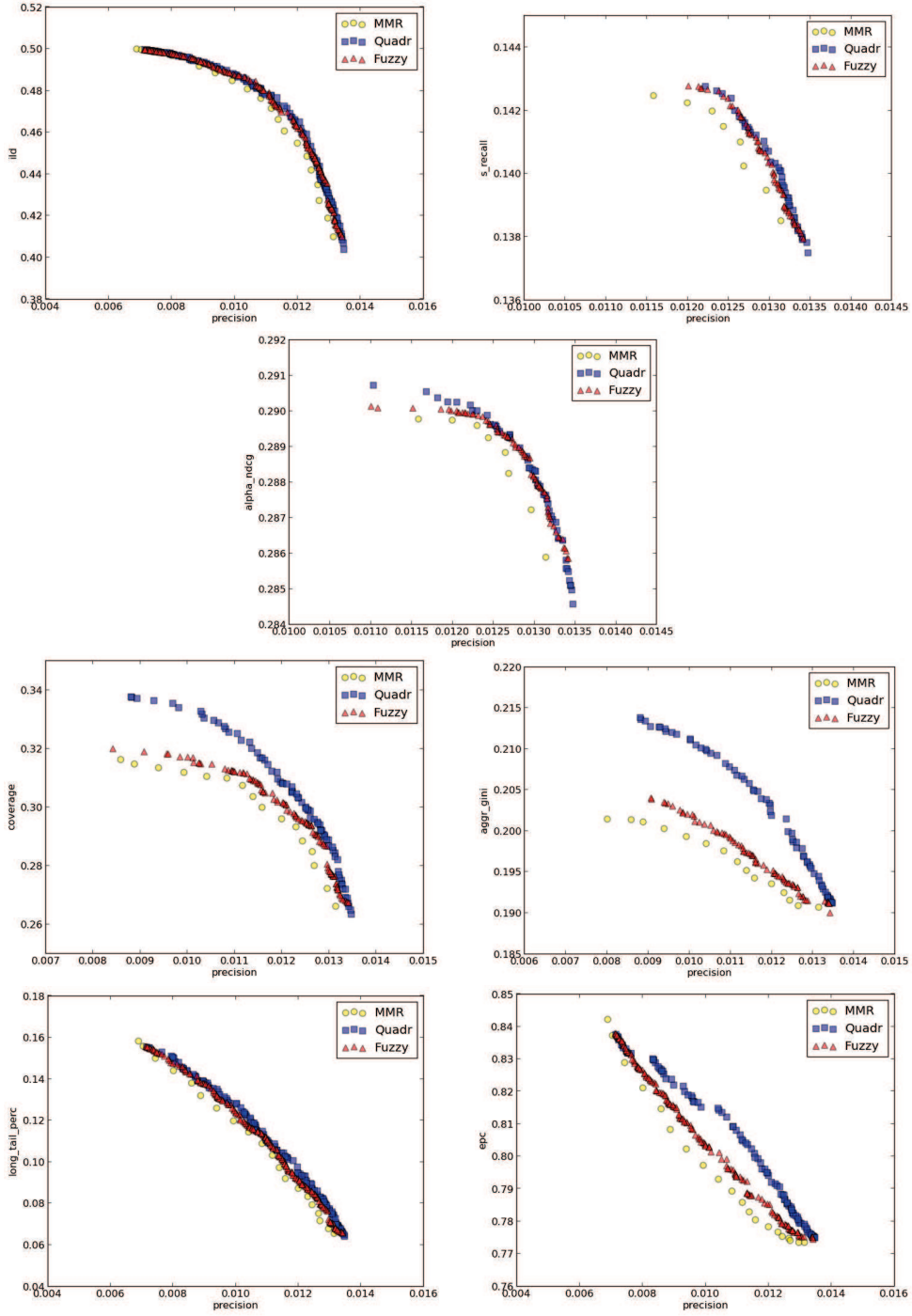


Figure 3.6: Pareto Frontiers for `LibraryThing` Dataset, using *BPRSLIM* and `MMR`

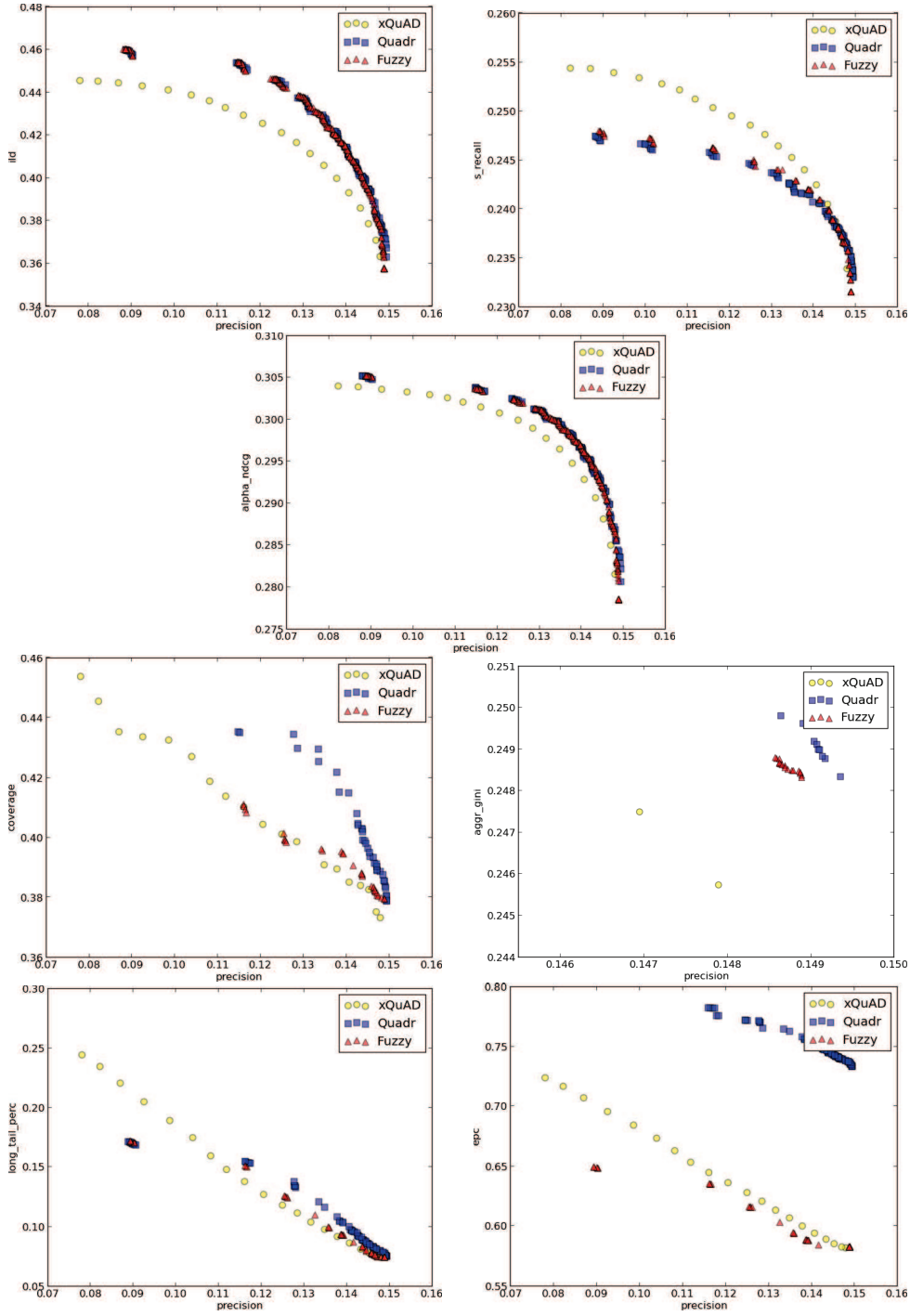


Figure 3.7: Pareto Frontiers for `Movielens` Dataset, using *BPRSLIM* and `xQuAD`

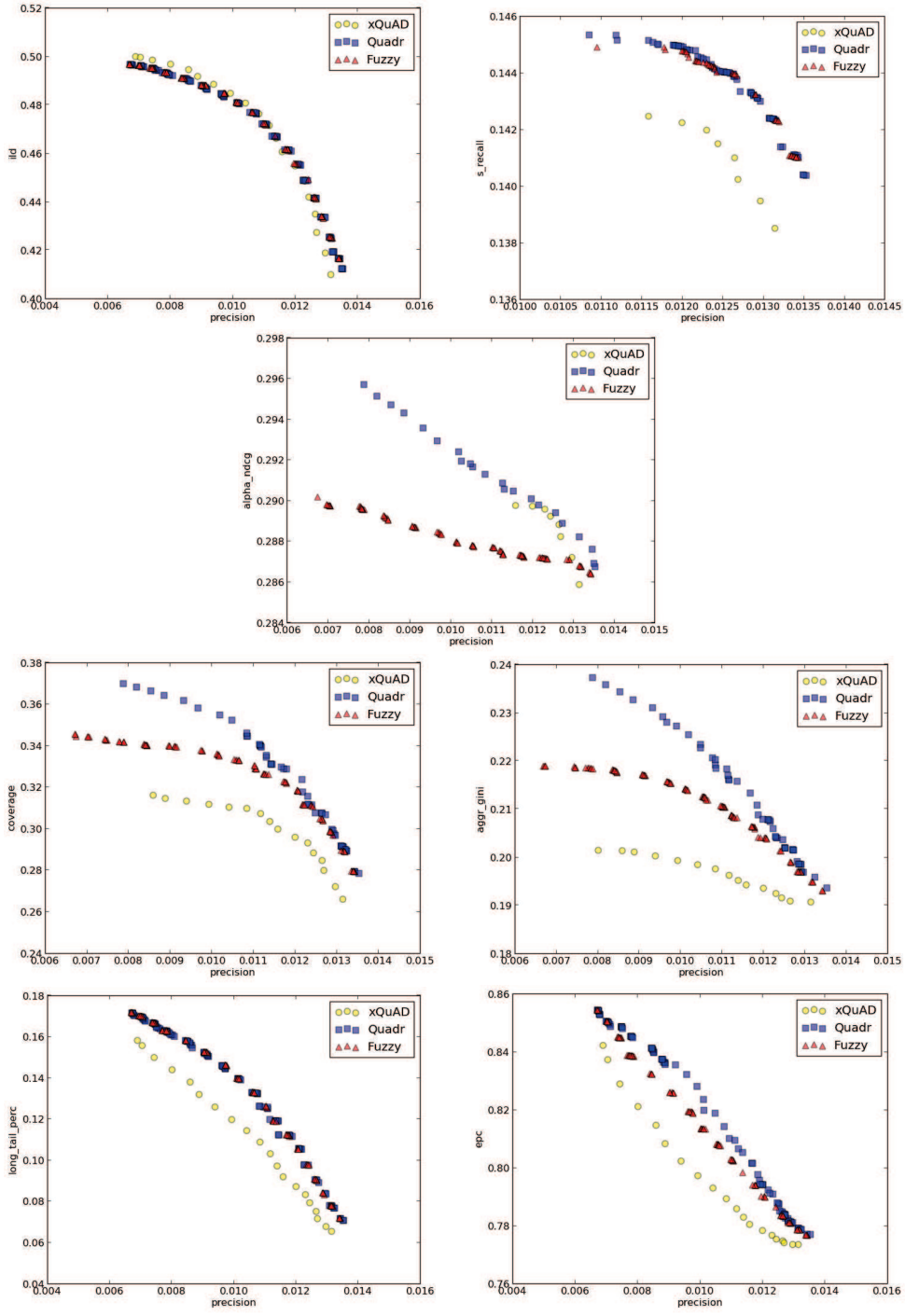


Figure 3.8: Pareto Frontiers for `LibraryThing` Dataset, using *BPRSLIM* and `xQuAD`

Chapter 4

Regression Trees for Intent-aware Multi-attribute Diversity

Analogously to the modelling and exploitation of query intent in Information Retrieval adopted to improve diversity in search results, in this chapter we focus on eliciting and using the profile of a user which is in turn exploited to represent her intents. The model is based on regression trees and is used to improve personalized diversification of the recommendation list in a multi-attribute setting. We tested the proposed approach and showed its effectiveness in two different domains, i.e. books and movies.

4.1 Introduction

In the recent years, diversification has gained more and more importance in the field of recommender systems. Engines able to get excellent results in

terms of accuracy of results have been proved to be not effective when we consider other factors related to the quality of user experience [98]. As a matter of fact, when interacting with a system exposing a recommendation service, the user perceives as good suggestions those showing also an appropriate degree of diversity, novelty or serendipity, just to cite a few. The attitude of populating the recommendation list with similar items could exacerbate the over-specialization problem that content-based recommender systems tend to suffer from [69], even though it appears also in collaborative-filtering approaches. Improving diversity is generally a good choice to foster the user satisfaction as it increases the odds of finding relevant recommendations [3]. Here our focus is on both the *individual* diversity and *aggregate diversity* (see Section 2.9.4 for more details) The item-to-item dissimilarity can be evaluated by using content-based attributes (e.g. genre in movie and music domains, product category in e-commerce) [156] or statistical information (e.g. number of co-ratings) [169]. Usually, approaches to the diversification take into account only one single attribute while, in the approach we present here, multiple attributes are selected to describe the items. The rationale behind this choice is that we believe there are numerous and heterogeneous item dimensions conditioning user's interests and choices. Moreover, depending on the user these dimensions may interact with each other thus contributing to the creation of her intents. The question is how to tackle multiple attributes to address the diversification problem.

In this work we use regression trees as user modeling technique to infer the individual interests, useful to provide an intent-aware diversification. Compared to approaches where item attributes are treated independently one to each other, regression trees make possible to represent user tastes as a combination of interrelated characteristics. For instance, a user could have a preference for horror movies of the 80s irrespective of the director, or for horror movies of the 90s directed by a a specific director. In a regression tree, conditional probability lets to build such inference rules about user's preferences. We conducted experiments on the movie and on the book domains to

empirically evaluate our approach. The performance was measured in terms of accuracy and both individual and aggregate diversity.

Specifically, we identify two main research questions:

- **RQ1** ***How do regression trees can be used to model user intents across multiple attributes in the diversification process?***
- **RQ2** ***How beneficial in terms of accuracy and diversity is to exploit regression trees for user intents modeling in a intent-aware diversification method?***

The main contributions of this chapter are:

- the presentation of a novel intent-aware diversification approach able to combine multiple attributes. It bases on the use of regression trees (and rules) to infer and encode the model of users' interests;
- the presentation of a novel method to combine different diversification approaches;
- an experimental evaluation which shows the performance of the proposed approaches with respect to both accuracy and diversity measures.

4.2 Related work

As shown in the previous chapter, multi-attribute diversity has been substantially non-treated in the literature of recommender systems.

The authors of [34] have highlighted a causal relationship between personality factors (such as openness and conscientiousness) and the degree of diversification in the user choices with respect to different attributes in the movie domain - genres, actors, directors, country or year of release. Considering this enlightenment, we can assert that different users can have a different degree of interest in diversification across different attributes. Please refer to

the previous Chapter for more detailed information about the multi-attribute diversification problem.

Furthermore, increasing attention has been paid to the intent-aware diversification, namely the process of increasing the diversity taking into account the user interests (also known as *intents*). To date, different approaches have been proposed for diversifying the recommendations, as shown in Section 2.9, but none of them considers more that attribute in the intent modeling, except for the approach described in this Chapter, whose results have been published in [153].

4.3 Intent-aware Multi-attribute Diversity

In this section we show how we address the intent-aware diversity problem when dealing with multi-attribute item descriptions. We used an adaptation of `xQuAD`, that allows to deal with the multi-attribute problem, proposed in the previous Chapter and explained in the Section 3.3.4. Here we refer to features as possible instances of a generic attribute.

Besides dealing with multi-attribute descriptions, the idea behind our approach is to infer and model the user profile by means of a regression tree, a predictive model where the user interest represents the target variable, which can take continuous values. Once a regression tree is produced for a user u, then it is converted into a set of rules $RT(u)$. Each rule maps the presence/absence of a categorical feature or a constraint on a numerical one to a value v in a continuous interval. This latter indicates the predicted interest of the user on the items satisfying the rule. In our implementation we used the interval $[1, 5]$ since the value of the target variable has been calculated as the rating mean of the training instances classified by the inferred rule. Please note that the choice of a specific value interval for the target variable does not affect the overall approach. Each rule m has then the form

$$body(m) \mapsto interest = v$$

with $body(m) = \{c_1, \ldots, c_n\}$. An example of a set of rules produced for a user is shown in Figure 4.1.

1. $\{horror \in dom(genres),\ western \notin dom(genres),$
   $DarioArgento \in dom(directors)\} \mapsto interest = 4.2$

2. $\{horror \notin dom(genres),\ thriller \in dom(genres)\}$
   $\mapsto interest = 2.1$

3. $\{year > 1990,\ horror \notin dom(genres),$
   $drama \in dom(genres),\ Aronofsky \in dom(directors)\}$
   $\mapsto interest = 4.0$

4. $\{year < 1990,\ drama \in dom(genres),$
   $AlPacino \in dom(actors)\} \mapsto interest = 3.9$

5. $\{horror \notin dom(genres)\} \mapsto interest = 3.2$

Figure 4.1: **Example of a set of rules generated via the regression tree**

Eventually, under the assumption that they represent specific user interests, the computed rules are used in the re-ranking phase as item features to improve the intent-aware recommendation diversity.

We propose also a *div* function for xQuAD so that each item is evaluated according to the rules it satisfies.

$$div^{rules}(i, \mathbf{S}, u) = \sum_{m \in M(u,i)} p(m|u)(1 - \text{avg}_{j \in \mathbf{S}}\, p(j|m)) \tag{4.1}$$

Here $M(u, i)$ represents the set of rules for the user u matched by the item i while $p(m|u)$ represents the importance of the rule m for u and is computed as:

$$p(m|u) = \frac{interest_m}{|M(u, i)|} \tag{4.2}$$

In Equation 4.2, $interest_m$ is the normalized predicted outcome of the regression tree for the rule m. Finally, the last component in Equation 4.1 indicates the complement of the coverage of the rule among the already se-

lected recommendations. We propose two different versions of this adapted xQuAD.

- **RT.** $p(j|m)$ is a binary function that returns 1 if the item j matches the rule, 0 otherwise.

- **DivRT.** $p(j|m)$ is the average similarity between m and each rule covered by item j. More formally:

$$p(j|m) = \text{avg}_{m' \in M(u,j)}\, sim(m, m') \tag{4.3}$$

The rationale behind this formulation is that some rules may be similar with each other thus not bringing any actual diversification if considered separately. The computation of $sim(m, m')$ takes into account the overlapping between the rules m and m' as follows:

$$sim(m, m') = \frac{\sum_{c_i \in body(m)} overlap(m, m', c_i)}{max(|body(m)|, |body(m')|)}$$

For instance, considering the attributes represented in Figure 4.1, we have for $actor, genre$ and $director$:

$$overlap(m, m', c_i) = \begin{cases} 1, & c_i \in body(m) \wedge c_i \in body(m') \\ 0, & \text{otherwise} \end{cases}$$

For the numerical attribute $year$ we may adopt a different formulation for the function $overlap(m, m', c_i)$. Here we compute, if any, the overlap between the interval in $body(m)$ and the one in $body(m')$ normalized with respect to maximum interval's length. As an example, if $year > 1990$ is in $body(m)$ and $year < 2010$ is in $body(m')$ we may define the overlapping function as $overlap(m, m', c_i) = \frac{|1990-2010|}{max(dom(year))-min(dom(year))}$.

The functions introduced above have been used in the experimental setting in order to compute the function $overlap(m, m', c_i)$ (see Section 4.4).

RT and DivRT can be used instead of the basic xQuAD as diversification algorithms in the re-ranking phase. Alternatively, basic xQuAD and RT or DivRT can be pipelined to benefit from the strengths of them both. For instance, one could use xQuAD to select 50 diversified recommendations and then RT to select 20 recommendations from those 50, or vice versa. Hereafter, we use the syntax X-after-Y, e.g. xQuAD-after-RT, to indicate that algorithm X is executed on the results of Y.

4.4 Experimental setting

We carried out a number of experiments to evaluate the performance of the methods presented in the Section 5.3 on two well known datasets: MovieLens1M and LibraryThing.

`MovieLens 1M`[1] dataset contains 1 million ratings from 6,040 users on 3,952 movies. The original dataset contains information about genres and year of release, and was enriched with further attribute information such as actors and directors extracted from DBpedia[2]. More details about this DBpedia enriched version of the dataset are available in [112]. Because not all movies have a corresponding resource in DBpedia, the final dataset contains 998,963 ratings from 6,040 users on 3,883 items. We built training and test sets by employing a 60%-40% temporal split for each user.

Moreover, we used the `LibraryThing`[3] dataset, which contains more than 2 million ratings from 7,279 users on 37,232 books. As in the dataset there are many duplicated ratings, when a user has rated more than once the same item, we selected her last rating. The unique ratings are 749,401. Also in this case, we enriched the dataset by mapping the books with BaseKB[4], the RDF version of Freebase[5] and then extracting three attributes: *genre*, *author*

[1] Available at `http://grouplens.org/datasets/movielens`
[2] `http://dbpedia.org`
[3] Available at `http://www.macle.nl/tud/LT`
[4] `http://basekb.com`
[5] `https://www.freebase.com`

and *subjects*. The subjects in Freebase represent the topic of the book, for instance Pilot experiment, Education, Culture of Italy, Martin Luther King and so on. The dump of the mapping is available online[6]. The final dataset contains 565,310 ratings from 7,278 users on 27,358 books. We built training and test sets by employing a 80%-20% hold-out split. The different ratio used for LibraryThing respect to Movielens (60%-40%) depends on its higher sparsity: holding 80% to build the user profile ensures a sufficient number of ratings to train the system.

For both datasets, we used the Bayesian Personalized Ranking Matrix Factorization algorithm (BPRMF) available in MyMediaLite[7] as baseline (using the default parameters). We performed experiments using other recommendation algorithms, whose results are shown in the Appendix C.

We selected the top-200 recommendations for each user to generate the initial list P used for performing the re-ranking as shown in Algorithm 1.

Accuracy is measured in terms of Precision, Recall and nDCG, but we only report nDCG values since the trend of the other two metrics is very similar. Individual diversity is measured using ILD and α-nDCG with $\alpha = 0.5$ to equally balance diversity and accuracy, while aggregate diversity is measured using both the catalog coverage and the entropy (see Section 2.9.4).

As similarity measure for computing the ILD metric (Equation 2.21) we used the Jaccard index. Considering that there are more attributes for each item, we computed the average of the Jaccard index value for each attribute shared between two items. α-nDCG is computed as the average of the Equation 2.23 for each attribute.

As presented in Section 5.3, we propose two novel diversification approaches: RT and DivRT. We also propose a method to combine in sequence different algorithms by means of a two phase re-ranking procedure, with the aim of benefiting from the strengths of both. Therefore we evaluated other two approaches: xQuAD-after-RT and RT-after-xQuAD, applying the sec-

[6] http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/BaseKB2LibraryThing.zip

[7] http://mymedialite.net/

ond re-ranking phase on the set of 50 recommendations provided from the first phase. We have also evaluated the combination with xQuAD and DivRT, but the results are very similar using RT, so they will not be shown. To evaluate the performances, we compare the top-10 recommendation list generating from all the approaches with basic xQuAD, by varying the λ parameter from 0 to 0.95 with step fixed to 0.05 in Equation 2.19 (higher values of λ give more weight to accuracy, lower values to diversity).

The rules are produced using M5Rules[8] algorithm available in Weka based on the M5 algorithm proposed by Wang and Witten [163]. M5Rules generates a list of rules for regression problems using a separate-and-conquer learning strategy. Iteratively it builds a model tree using M5 and converts the best leaf into a rule. We decided to use unpruned rules in order to have more rules matchable with the items.

4.5 Results Discussion

Results of the experiments on MovieLens and LibraryThing are reported in Figure 4.2 and 4.3, respectively.

MovieLens. xQuAD obtains the best results in terms of ILD (Figure 4.2(a)) and α-nDCG (Figure 4.2(b)), though the xQuAD-after-RT results are very close and, with higher λ values (namely giving more importance to the accuracy factor), the differences between them are not significant. This outcome is due to the fact that the diversity metrics are attribute-based and xQuAD operates directly diversifying the attributes values, while the proposed rule-based approaches do not take into account all the attributes values. This also explains why the pure rule-based approaches (RT and DivRT) obtain the worst diversity results, while the combined algorithms (xQuAD-after-RT and RT-after-xQuAD) obtain better results. It is noteworthy that these last two configurations have no substantial difference with

[8] http://weka.sourceforge.net/doc.dev/weka/classifiers/rules/M5Rules.html

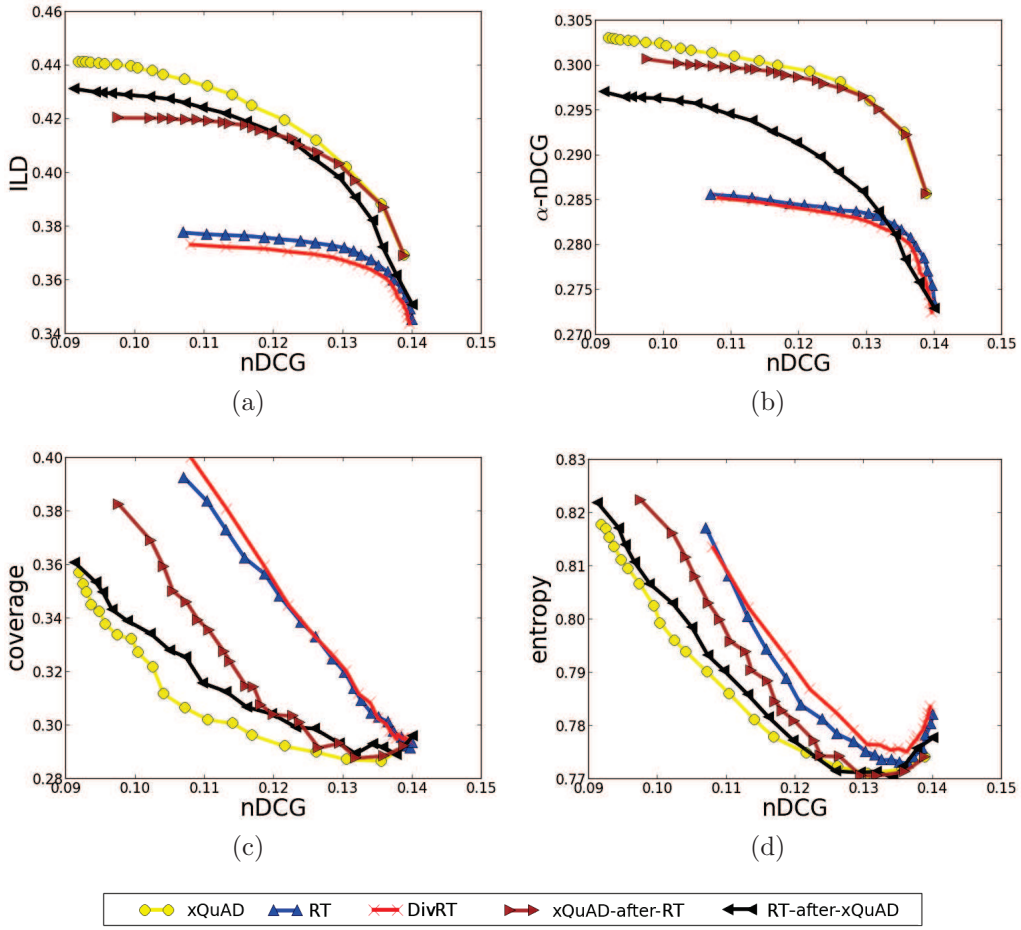


Figure 4.2: Accuracy-diversity curves on MovieLens at Top-10 obtained by varying the λ parameter from 0 to 0.95 (step 0.05). The statistical significance is measured based on the results from individual users, according the Wilcoxon signed-rank significance test. For nDCG and ILD 4.2(a), all the differences are statistically significant with ($p < 0.01$), except for those between RT and DivRT. For α-nDCG 4.2(b), the trend is the same, except for the differences between xQuAD and xQuAD-after-RT with $\lambda > 0.7$.

ILD, but, in terms of α-nDCG, xQuAD-after-RT considerably overcomes RT-after-xQuAD. This demonstrates that the pipeline of xQuAD and the rule-based approach obtains good diversity. Considering coverage (Figure 4.2(c)) and entropy (Figure 4.2(d)) to evaluate the aggregate diversity, the results

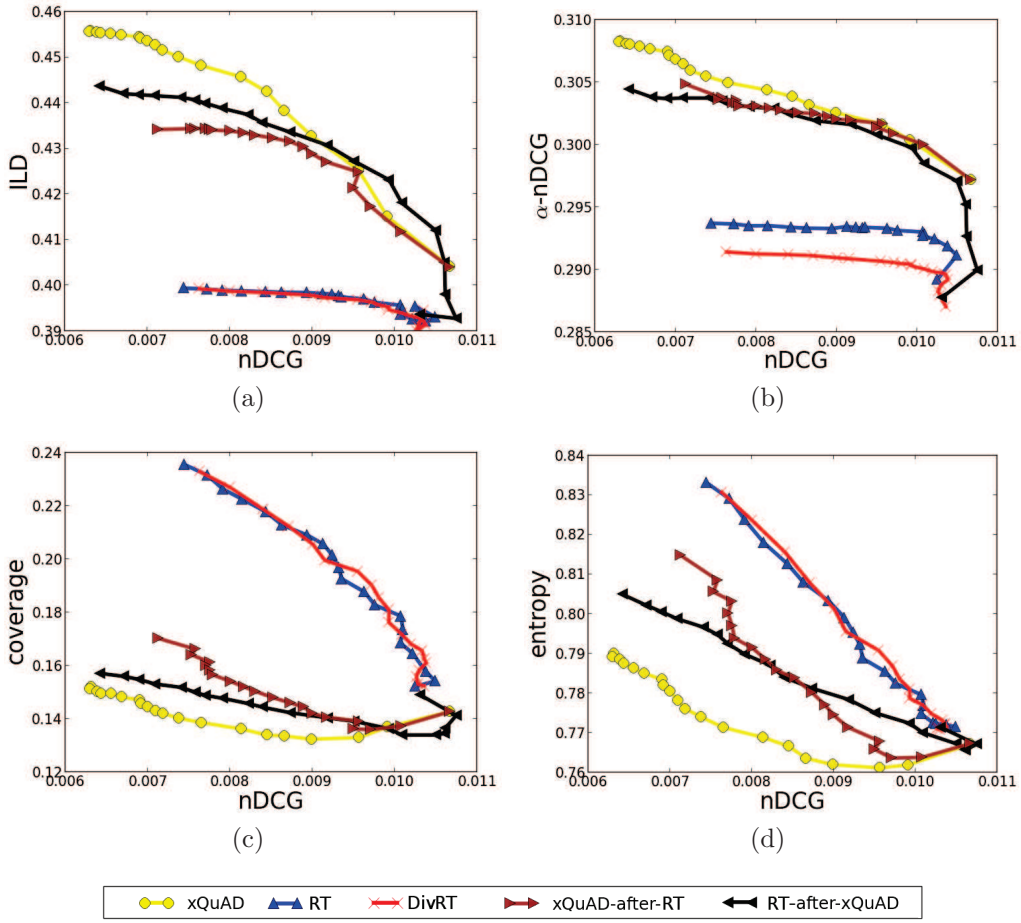


Figure 4.3: Accuracy-diversity curves on LibraryThing at Top-10 obtained by varying the λ parameter from 0 to 0.95 (step 0.05). The statistical significance is measured based on the results from individual users, according the Wilcoxon signed-rank significance test. For nDCG, the differences between RT and DivRT are non significant with $\lambda \in [0.2, 0.5]$. For ILD 4.3(a), all the differences are statistically significant with ($p < 0.001$), except for those between RT and DivRT. For α-nDCG 4.3(b), all the differences are statistically significant ($p < 0.001$).

show that using the rules the recommendations are much more personalized. It is interesting to note the compromise provided by xQuAD-after-RT, that obtains equidistant results between xQuAD and the rule-based algorithms,

unlike RT-after-xQuAD that slightly overcomes xQuAD. With respect to the baseline, no configuration is able to give more accurate recommendations (nDCG = 0.14); all are able to increase the individual diversity (ILD = 0.34 and α-nDCG = 0.27). With nDCG and the individual diversity, the differences are always statistically significant ($p < 0.001$), except using the pure ruled-based approaches with $\lambda > 0.65$. The situation is more complex in terms of aggregate diversity, since the coverage grows very little on the baseline (coverage = 0.29) and the entropy slightly decreases (entropy = 0.78) with higher λ values. According to a comprehensive analysis on MovieLens, the pure rule-based approaches may give personalized and diversified recommendations, also with small accuracy loss. However, when individual diversity is more important than aggregate diversity, combining xQuAD with a previous rule-based re-ranking gives a good compromise between individual and aggregate diversity.

LibraryThing. At first glance, the LibraryThing results appear similar to those on MovieLens. Although they are generally consistent, there are interesting differences. Also in this case, xQuAD obtains the best diversity values, with ILD (Figure 4.3(a)) and α-nDCG (Figure 4.3(b)). However, both the combined approaches obtain really interesting results, very close to xQuAD, except for the lower λ values (namely giving more importance to the diversification factor). Unlike what happens on MovieLens, in this case RT-after-xQuAD obtains good results also in terms of α-nDCG. The pure rule-based approaches still obtain worse results. Considering coverage (Figure 4.3(c)) and entropy (Figure 4.3(d)) to evaluate the aggregate diversity, the results show that using the rules the recommendations are much more personalized than using only xQuAD. The combined approaches are able to improve the aggregate diversity with respect to xQuAD, albeit they are still distant from the pure rule-based approaches, especially in terms of coverage. With respect to the baseline, all configurations give a little more accurate recommendations, with $\lambda > 0.65$, but the differences are not statistically significant. In terms of individual diversity, all of them are able to overcome

the baseline (ILD = 0.4 and α-nDCG = 0.285) except when using the pure rule-based approaches in terms of ILD. However they are able to improve α-nDCG. For the latter two metrics, the differences are always statistically significant ($p < 0.001$). In terms of aggregate diversity, xQuAD does not improve the baseline result (coverage = 0.15 and α-nDCG = 0.77), while using the rules leads to better results. According to a comprehensive analysis on LibraryThing, the pure rule-based approaches may give more personalized recommendations with a better diversity, especially using RT, with also a small accuracy loss. Similarly to the analysis on MovieLens, the results on LibraryThing suggest that diversifying with only the rules is a good choice when aggregate diversity is more important than individual diversity, conversely xQuAD remains the best choice to improve the individual diversity and combined with the rule-based diversification improves also the aggregate diversity.

The final conclusions of this analysis are that using a regression tree to infer rules representing user interests on multi-attribute values in the diversification process with xQuAD leads to more personalized recommendations but with a less diversified list and that combining attribute-based and rule-based diversifications in two phase re-ranking is a good way for taking the advantages of both. The better degree of personalization may depend on the fact that the rules are different among the users since they represents their individual interests. The lower individual diversity values with ILD and α-nDCG are due to the nature of these metrics which are based directly on the attributes values while the pure rule-based approaches do not take into account all the attributes values.

4.6 Summary

This chapter addresses the problem of intent-aware diversification in recommender systems in multi-attribute settings. The proposed approach bases on

xQuAD [159], a relevant intent-aware diversification algorithm, and leverages regression trees as user modeling technique. In their rule-based equivalent representation, they are exploited to foster the diversification of recommendation results both in terms of individual diversity and in terms of aggregate one. The Section 4.3 describes our proposed method to use regression trees for user intents modeling, responding to the question RQ1.

The experimental evaluation on two datasets in the movie and book domains, as an answer to question RQ2, demonstrates that considering the rules generated from the different attributes available in an item description provides diversified and personalized recommendations, with a small loss of accuracy. The analysis of the results suggests that a pure rule-based diversification is a good choice when the aggregate diversity is more needed than individual diversity. Conversely, basic xQuAD remains the best choice to improve the individual diversity while its combination with the rule-based diversification improves also the aggregate diversity.

Chapter 5

Diversification with Temporal Dynamics

In this chapter we propose the analysis of temporal dynamics for a better user intent modeling. In particular, we propose a time-based analysis relying on a temporal decay function and another method based on a new technique for session analysis. We applied the proposed methods as intent model in an intent-aware diversification framework called xQuAD and evaluated them in terms of accuracy, individual diversity and aggregate diversity.

5.1 Introduction

It has been strongly demonstrated the importance of analyzing temporal dynamics for user modeling [80, 81, 72]. Inspired by such works, here we focus on a temporal analysis of user activities for improving the trade-off between accuracy and diversity of the recommendation lists. The intuition behind our idea is that temporal dynamics might allow to better understand

the user interests with respect to the items characteristics and then provide a more accurate intent-aware diversification. Therefore, this work presents two intent modeling methods based on temporal dynamics. The first one analyses the frequency of interaction between the users and the items features using a temporal decay function in order valorize *persistence* and *recency* of an intent. The other method is based of a new session analysis technique of user ratings for intent modeling. Considering that a session is usually defined as a set of consecutive ratings with a very small gap of time among them (e.g. less than one hour in music [80]), we provided a wide definition of session tailored for movie ratings. In particular, such method is designed to valorize *importance*, *persistence* and *recency* of an intent among the user sessions. We have experimentally evaluated such methods with the large scale movie dataset published in the context of Netflix Prize Context [19].

The main research questions we address in this chapter are:

- **RQ1** ***How beneficial in terms of accuracy and diversity is to exploit temporal dynamics for intent modeling in intent-aware diversification methods?***
- **RQ2** ***What is the best choice between the different methods proposed in this work?***

The evaluation aimed at evaluating the benefit of exploiting temporal dynamics for intent modeling in intent-aware diversification, and to determine whether the new method for sessions analysis proposed here can provide better results. Therefore, the focus of this work is on accuracy and individual diversity of the top-N recommendations. Along with such quality dimensions, we also considered the aggregate diversity to determine the personalization degree provided by the compared methods [7]. The experimental results demonstrated that the analysis of temporal dynamics leads to better accuracy-diversity balance and intent-aware diversity compared to the original `xQuAD`, but only using the session analysis technique. As additional benefit, the aggregate diversity results improved too thus demonstrating to

produce more personalized recommendations.

5.2 Related Work

Recently, the importance of taking into account the temporal dynamics in recommendation task has taken hold. A method to model user sessions in music domain was proposed in [80]. In particular, it considers as session each set of consecutive ratings without an extended time gap between them. Considering that there are vary psychological phenomena that lead to some ratings to be grouped in a single session, such method captures these effects by means of user session biases. [81] presented a collaborative filtering algorithm able to model time drifting of user preferences and the results on the Netflix dataset indicated the importance of uncovering temporal effects for the producing more accurate recommendations. Another method proposed to take advantage of temporal information of user behaviour is called Time-based Markow Embedding [72], used to find the best next-song recommendation via Latent Markov Embedding.

This work aims to explore the exploitation of uncovering temporal dynamics of user intents in order to provide a better intent-aware diversification. To the best of out knowledge, there are no previous work with the same purpose. Please refer to the Section 2.9 for more information about the intent-aware diversification problem.

5.3 Intent modeling with Temporal Dynamics

When temporal information about ratings is available, it is possible to model temporal dynamics of user activities. We proposed two methods to exploit temporal analysis for intent modeling in diversification that we call **time-based** and **session-based intent modeling**. Both relies on the intuition that user intents can change during the interaction with the systems and

evaluating the importance of a feature merely computing its frequency in the user profile may not represent the current user interests.

5.3.1 Time-Based Intent Modeling

In order to valorize *persistence* and *recency* of an intent, we propose to analyze the frequency of interaction between the user u and the feature f and to weight each interaction by a temporal decay function. More formally, the following formula computes the interest of the user u with respect to the feature f:

$$p(f|u) = \frac{\sum_{i \in R(u)} cov(f,i) disc(u,i)}{\sum_{i \in R(u)} disc(u,i)} \tag{5.1}$$

where $R(u)$ indicates the set of rating provided by the user u; $cov(f,i)$ is a binary function returning 1 if the item i is associated with the feature f, otherwise 0; $disc(i,u)$ is a temporal decay function returning lower values for older ratings, and higher values for the most recent ones.

Inspired by [81], as decay function we adopted the following exponential function

$$disc(u,i) = e^{-\beta \cdot |t_{u,last} - t_{u,i}|} \tag{5.2}$$

where $t_{u,last}$ indicates the date of the last rating of the user u and $t_{u,i}$ the date when user u rated i; $\beta > 0$ controls the decay rate. We tried several values of β and finally empirically chose 1/200. Moreover, as an alternative, we also proposed a different version of Equation (5.2), using a positive exponent and an opposite calculation of the temporal difference. More formally:

$$disc_{pos}(u,i) = e^{\beta \cdot |t_{u,i} - t_{u,first}|} \tag{5.3}$$

where $t_{u,first}$ indicates the date of the first rating of the user u. It is important to note that changing the sign of the exponent requires to change also the temporal difference interval in order to give more importance to more recent ratings and penalize older ones. The intuition behind the adoption of this latter function relies on an empirical attempts to obtain better results

during the evaluation step. Indeed, as showed in Section 5.5, such function demonstrated to be the best solution, in particular when applied to the session-based method (see next session).

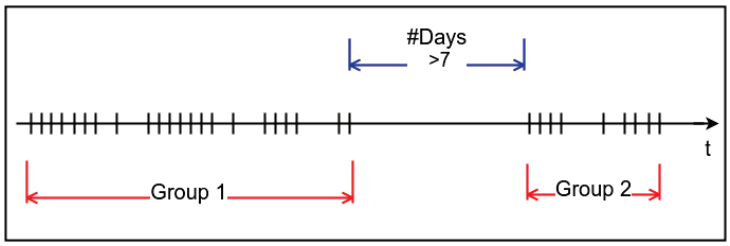
5.3.2 Session-Based Intent Modeling

User sessions definition

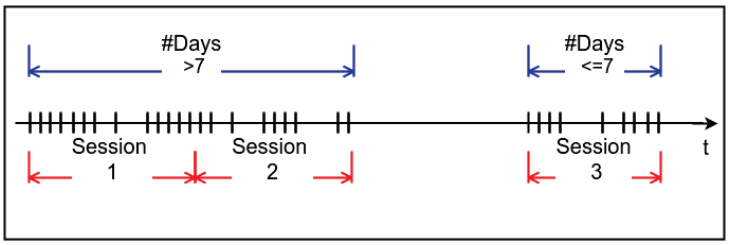
Session analysis is quite common in music domain, since users are used to listen to many songs in sequence. Therefore a session is represented by a set of consecutive ratings with a small gap of time among them [80]. Conversely, sessions are not easy to find in movie domain, since users typically watch a small number of movies in brief timeslots and the temporal gap among visions or ratings could be large (sometimes several days or even months). Therefore, we propose in this work a new method for splitting the ratings in sessions by means of the following steps:

1. We group the ratings with a time of inactivity shorter than 7 days (a week). Therefore, each group will contain a sequence of ratings relatively close. Figure 5.1(a) shows an example of this first phase, where the ratings are divided in two groups due to an inactivity interval longer than 7 days between them.

2. We split each of the previous group in a number of sub-groups, in order to have final sessions composed by a number of days from 5 to 10. Then, we fix the number of sub-groups equals to the round of the number of days in the initial group divided by 7. Figure 5.1(b) shows how the groups in Figure 5.1(a) are divided in sessions. As we can see, the first group lasts more than 7 days and is hence divided in two final sessions, while the second group becomes a whole session.



(a)



(b)

Figure 5.1: User profile splitting in sessions in two phases. Figure 5.1(a) shows how to split the ratings into groups separated by an inactivity period longer than 7 days. Figure 5.1(b) represents the transformation of each group in one or more final sessions.

Intent modeling

Once user sessions are determined, they can be used to analyze the user activities taking into account the temporal dynamics. In this work we present an approach to model the users intents over time, by considering three key properties: *importance*, *persistence* and *recency* of an intent among the user sessions. The first property indicates the importance of an intent in each session computed as the percentage of items covering that intent. The second property considers how many sessions the intent is important for, therefore it sums the importance of the intent for each sessions. Finally, the third property focuses on the intent freshness, penalizing old sessions with a temporal decay function.

More formally, the following formula computes the interest of the user u with respect to the feature f:

$$p(f|u) = \frac{\sum_{s \in sessions(u)} \frac{\sum_{i \in I(s)} cov(f,i)}{|F(s)|} disc(s,u)}{\sum_{s \in sessions(u)} disc(s,u)} \tag{5.4}$$

where *sessions(u)* indicates the set of sessions found in the user profile of u; *I(s)* is the set of items in the session s; *cov(f,i)* is a binary function returning 1 if the item i is associated with the feature f, otherwise 0; *F(s)* represents the set of features associated with all the items in the session s; *disc(s)* is the temporal decay function adapted to handle the sessions instead of the items, considering a session as an item in Equation (5.2) or (5.3) where the session date is that of the last rated item in such session. Again, we empirically fixed β to 1/200.

5.4 Experimental setting

Dataset. In order to verify our research questions and evaluate our proposal, we used the movie dataset derived from the Netflix Prize Context [19]. It originally contains over 100 million ratings provided by ˜480,000 users on ˜17,000 movies. Such ratings were collected between 1998 and 2005 and associated with the relative date. However, such dataset contains noise added on purpose for reasons of privacy, as explained in the Netflix Prize Rules[1]: "some of the rating data for some customers in the training and qualifying sets have been deliberately perturbed in one or more of the following ways: deleting ratings; inserting alternative ratings and dates; and modifying rating dates". Indeed, we found that some users rated an exaggerated number of movies in some days: 30% of all the users have rated at least 61 movies in the most *prolific* day. Therefore we selected a sample of users removing those considered as outliers by means of the following steps:

1. We discarded the users with less than 20 ratings.
2. We ordered the users in decreasing order of the maximum number of daily interactions and discarded the top 30%.

[1] http://netflixprize.com/rules.html

3. We repeated the previous step considering the average number of daily interactions instead of the maximum.

The final dataset contains 233,452 users, 18,104,476 rating and 17,763 movies. We built training and test sets by employing a 70%-30% temporal split for each user. In order to extract the genre information, we mapped each movie with the corresponding Freebase resource by means of its title and year of release. Overall, the number of distinct genres extracted is 266.

Recommendation algorithms. As baselines we selected two state-of-the-art collaborative filtering algorithms available in MyMediaLite[2]: `BPRMF` and `BPRSLIM`. Note that for each baseline a separate evaluation is required. Given a baseline, it is used to produce for each user a list of 300 recommendations that represents the initial list $\overline{\mathbf{R}}$ in Algorithm 1. In other words, it is used to compute $r^*(u, i)$ in Equation (2.19). We used `xQuAD` as diversification algorithm (see Section 2.9.1 for more details).

The time-based and session-based intent modeling proposed in Section 3.3 can be used as alternatives to the classic frequency based intent modeling in the original `xQuAD`. In turn, for both such methods, the temporal decay can be computed using Equation (5.2) or (5.3), as shown in Section 3.3. Therefore, combining the two methods with the two formulas we obtained four variations of `xQuAD`, denoted as: `TBn_xQuAD`, `TBp_xQuAD`, `SBn_xQuAD`, `TBp_xQuAD`, where *TB* stands for time-based and *SB* for session-based, while the third letter indicates whether the exponent in the decay formula is the negative (n) or positive (p) version, corresponding respectively to Equation (5.2) and (5.3).

Metrics. In order to evaluate the *accuracy*, we used Precision, Recall and nDCG. *Individual diversity*, namely the degree of dissimilarity among all items in the list provided to a user, was measured by two metrics: ILD (Intra-List Diversity) and ERR-IA (Intent-Aware version of ERR) [33]. ILD is the most frequently used metric and it only measures the diversity in a list, while ERR-IA takes into account relevance as well and it has been demonstrated

[2] http://www.mymedialite.net

that xQuAD implicitly targets ERR-IA [29]. We considered catalog coverage and entropy as *aggregate diversity* metrics (see Section 2.9.4).

5.5 Results Discussion

Charts in Figure 5.2 show the curves between Precision and ILD, and Precision and ERR-IA, of the different methods based on xQuAD using BPRMF as baseline. Figure 5.3 shows the same curves for BPRSLIM. The trends are substantially the same between BPRMF and BPRSLIM.

At a first glance, TBn_xQuAD and TBp_xQuAD result not able to improve neither accuracy or diversity, since both show a trend very similar to xQuAD but with a strongly reduced range. In particular, for all the values of λ their results do not differ significantly from xQuAD with λ from 0.9 to 0.6, considering both ILD and ERR-IA.

Conversely, the other two proposed methods, the ones based on session analysis, can overcome xQuAD in terms of diversity. Considering the ILD metric, SBp_xQuAD, namely the session-based method using the decay function with positive exponent (Equation (5.3)), yields the most diversified recommendations but with a substantial drop of accuracy. However, SBp_xQuAD can obtain similar accuracy results but still with better diversity compared to the other methods when high values of λ are used in the objective function (Equation (2.19)), recalling that higher values give more weight to the accuracy and less to diversity in the re-ranking phase. Therefore, SBp_xQuAD allows to increase the diversification preserving the accuracy, when adequate values of λ are selected, thus resulting the more convenient choice when diversity is the main objective. Such outcome is confirmed by the analysis of the ERR-IA results, where SBp_xQuAD provides a strong improvement over all the other methods. Again, using high values of λ, SBp_xQuAD overcomes all the other methods in terms of diversity without significant loss of accuracy.

It is worth to note that SBp_xQuAD with $\lambda = 0.9$, represented by the highest point in the chart, is already able to provide better intent-aware

diversification compared to the other methods independently by their values of λ. While, fixing $\lambda = 0.7$ such method provides the best overall ERR-IA with a slight loss of accuracy.

While `SBp_xQuAD` seems the best method overall, `SBn_xQuAD` shows a strange behaviour. Even though it obtains a good balance of Precision and ILD, its results in terms of ERR-IA are strongly penalized, obtaining the worst intent-aware diversity degree. Explaining this difference requires further investigation, that we leave it as future work.

Table 5.1 shows the results selecting for each re-ranker the value of λ which maximize ERR-IA. Such results demonstrate the superiority of `SBp_xQuAD` in terms of ILD and ERR-IA compared to all the other methods with a loss of accuracy little relevant respect to the other re-ranker methods. Moreover, `SBp_xQuAD` yields also better aggregate diversity in comparison to almost all the other methods.

5.6 Summary

In this chapter we investigate the role of temporal information while modeling a user profile in computing diversified recommendations. We propose two different time-dependent user modelings which take into account also the user rating history. One of the two proposed methods bases on a new session analysis technique by considering those periods where the user interacted in a more constant way with the system. As an answer to question RQ1, the experimental evaluation demonstrated that considering temporal dynamics by means of the session-based analysis leads to better accuracy-diversity balance and better intent-aware diversification. In particular, using high values of λ, our proposed method called `SBp_xQuAD` overcomes all the other methods in terms of diversity without significant loss of accuracy. This last outcome responds to the question RQ2 regarding the best choice among the different methods proposed in this Chapter.

		Accuracy			Diversity		Aggregate Diversity	
	λ	**Precision**	**Recall**	**nDCG**	**ILD**	**ERR-IA**	**Coverage**	**Entropy**
BPRMF		0.0310	**0.0380**	**0.0337**	0.7210	0.0131	0.2984	7.9514
+xQuAD	0.6	0.0309	0.0358	0.0327	0.7395	0.0189	0.2808	7.9207
+TBn_xQuAD	0.1	***0.0316***	0.0369	*0.0335*	0.7402	0.0186	0.2935	7.9641
+TBp_xQuAD	0.1	*0.0310*	0.0362	0.0328	0.7411	0.0189	0.2850	7.9277
+SBn_xQuAD	0.1	0.0298	0.0363	0.0317	**0.7758**	0.0153	**0.3119**	8.1445
+SBp_xQuAD	0.7	0.0297	0.0350	0.0313	**0.7716**	**0.0249**	0.3042	**8.9926**
BPRSLIM		**0.0356**	**0.0452**	**0.0420**	0.7390	0.0165	0.3931	9.7800
+xQuAD	0.6	0.0328	0.0403	0.0373	0.7457	0.0201	0.3832	9.6000
+TBn_xQuAD	0.1	0.0331	0.0404	0.0373	0.7479	0.0200	0.3905	9.6400
+TBp_xQuAD	0.1	0.0320	0.0392	0.0359	0.7488	0.0200	0.3840	9.5600
+SBn_xQuAD	0.2	0.0339	*0.0432*	0.0394	0.7661	0.0174	**0.4002**	**9.7900**
+SBp_xQuAD	0.7	0.0326	0.0407	0.0374	**0.7694**	**0.0261**	**0.4002**	9.7500

Table 5.1: Accuracy, diversity and aggregate diversity results for BPRSLIM and BPRMF. For each re-ranker, the value of λ is the one that maximize ERR-IA. The *italic* and underline fonts indicate that the difference is not statistically significant compared to the baseline and xQuAD algorithms, respectively; while all the other differences are statistically significant (Student's paired t-test rank with $p < 0.001$). Bold style indicates the best values in each column.

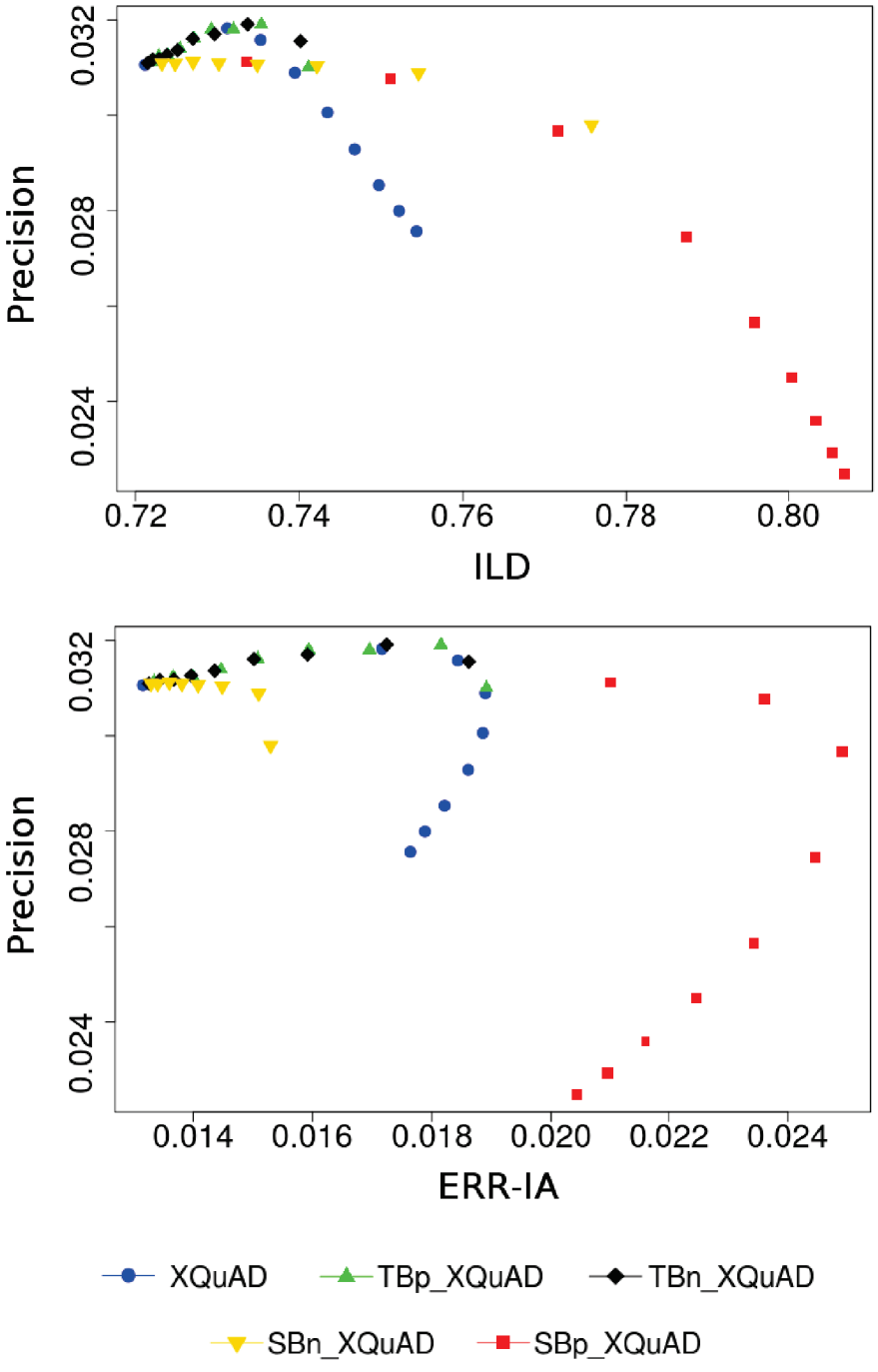


Figure 5.2: Curves obtained by varying the λ parameter from 0.9 to 0.1 (step 0.1), using `BPRMF` as recommendation algorithm. To better understand the curves, note that Precision and λ are directly proportional.

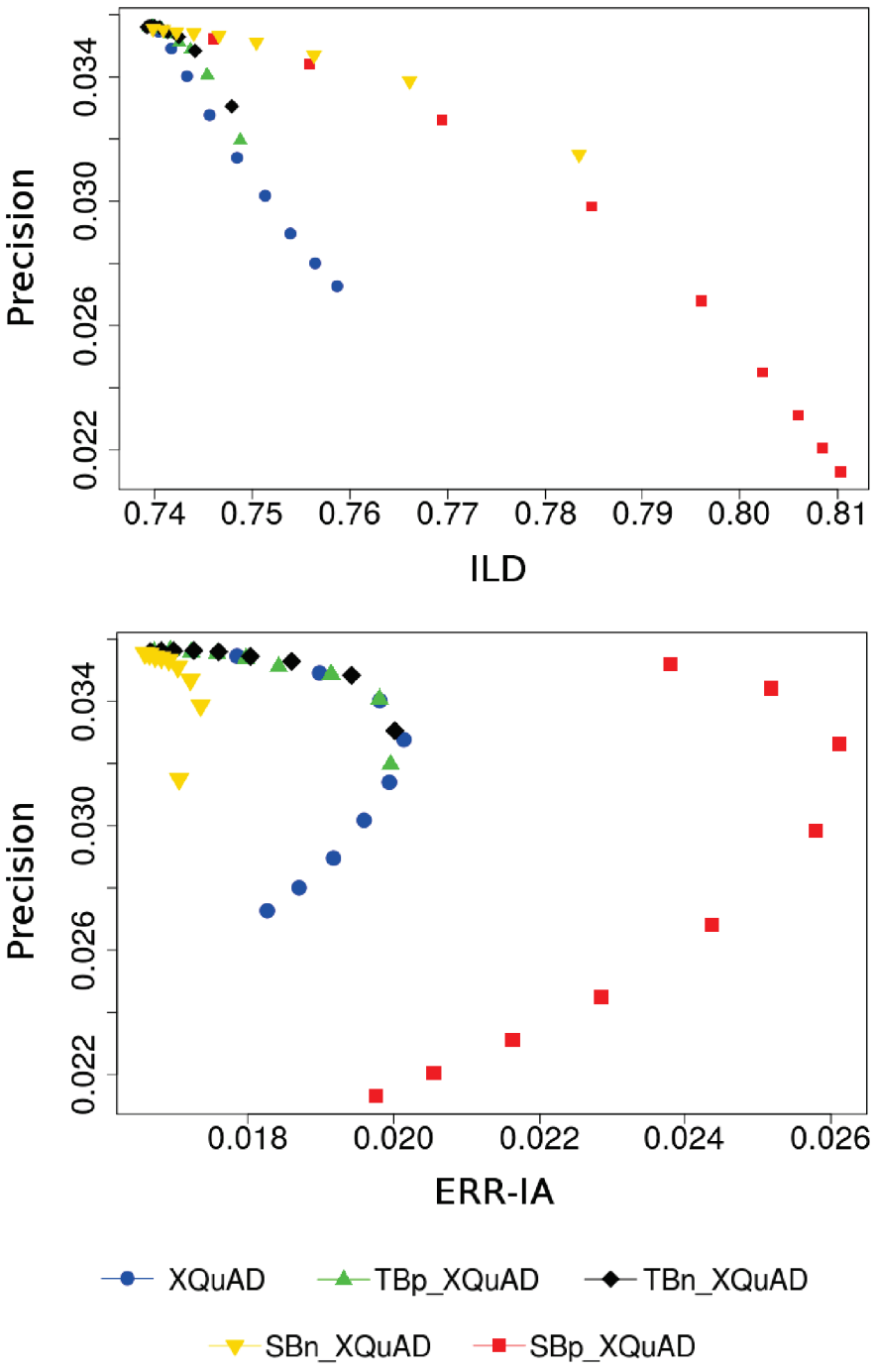


Figure 5.3: Curves obtained by varying the λ parameter from 0.9 to 0.1 (step 0.1), using `BPRSLIM` as recommendation algorithm. To better understand the curves, note that Precision and λ are directly proportional.

Chapter 6

Linked Data and Cross-Domain Recommendations For User Cold-Start Problem

Finding accurate recommendations for users in cold-start situations is a well-known challenge in the design of recommender systems. Often, additional data is used to compensate the scarcity of user feedback. In this work we address such problem in a target domain by exploiting user preferences from a related auxiliary domain. Following a rigorous methodology for cold-start, we evaluate a number of recommendation methods on a dataset with positive-only feedback in the movie and music domains, both in single and cross-domain scenarios. Comparing the methods in terms of item ranking accuracy, diversity and catalog coverage, we show that cross-domain preference data is useful to provide more accurate suggestions when user feedback in the target domain is scarce or not available at all, and may lead to more diverse recommendations depending on the target domain. Moreover, evaluating the

impact of the user profile size and diversity in the source domain, we show that, in general, the quality of target recommendations increases with the size of the profile, but may deteriorate with too diverse profiles.

6.1 Introduction

Providing relevant suggestions of items for new users is a well-known problem in recommender systems. In such cases there is little or no information about the users preferences, and traditional recommendation models are not able to compute meaningful personalized predictions.

Cross-domain recommender systems that leverage additional information from different but related source domains have been introduced as a potential solution to cold-start situations [26]. This auxiliary information can be exploited to mitigate the lack of historical data in the target recommendation domain, thus addressing the user cold-start [47]. In one of the first papers on the topic, Winoto and Tang [166] conjectured that although the introduction of cross-domain information could deteriorate the prediction performance in the general –non cold-start– case, it could still lead to more diverse recommendations. Subsequent work proposed methods to effectively learn and transfer knowledge from the source domain to the target [66], and found that the quality of the recommendations improves when the involved domains are semantically more related [133]. Nevertheless, to the best of our knowledge, no previous work has tested Winoto and Tang's conjecture regarding the diversity of recommendations when cross-domain data is exploited.

Moreover, in [44] it has been shown that users perceive differences in the recommendation quality depending on the variety of items they naturally prefer. Based on this observation, we hypothesize that both the amount and diversity of source domain preferences have an impact on the accuracy of cross-domain recommendations. Specifically, we identify three main research questions:

- **RQ1** ***How beneficial in terms of accuracy is to exploit cross-***

domain information for cold-start users? We analyze the ranking performance of the top-N recommendations based on positive-only feedback, following a principled evaluation methodology specifically designed for cold-start scenarios [78].

- **RQ2** ***Is cross-domain information really useful to improve the recommendation diversity?*** In order to test Winoto and Tang's conjecture we include in the evaluation the intra-list diversity metric and the recently proposed binomial diversity framework [157].
- **RQ3** ***What is the impact of the size and diversity of the user profile in the source domain on the quality of the target recommendations?*** We check this by computing the degree of diversity of the user profiles in the source domain. This work represents, to the best of our knowledge, the first analysis on user profile diversity for cross-domain recommendation.

We investigate these issues by evaluating a number of memory-based and matrix factorization algorithms in single and cross-domain scenarios, using two datasets with positive-only only feedback consisting of Facebook *likes* on movies and music artists, mapped to DBpedia[1] entities, whose metadata is used to also evaluate two state-of-the-art graph-based methods able to exploit heterogeneous information in the recommendation process.

6.2 Related Work

To compensate the lack of information in user cold-start problem, two major approaches have been studied in previous work: (i) preference elicitation techniques [132] that directly ask the user to provide some ratings before delivering recommendations, and (ii) methods that exploit additional information about the users to better estimate their preferences. In the latter case, some approaches combine content and collaborative information [139, 154],

[1] http://dbpedia.org

and others exploit demographic data or even the user's personality [49] to address the user cold-start problem. More recently, solutions based on cross-domain recommender systems have been explored with positive results [26], since auxiliary information extracted from related source domains allows to enrich the information in the target domain. Please refer to Section 2.10 for a more detailed overview of the possible solutions.

6.3 Dataset

The recommendation models presented in this work were evaluated on a Facebook dataset with user *likes* for movie and music items, which we extended with item metadata extracted from DBpedia. In [154] we detail the dataset and the developed process to automatically extract DBpedia semantic networks relating items and features. Next we provide a brief summary of them. In the original raw data –as acquired from the Facebook Graph API– each user-*like*-item relation was given as a 4-tuple with the identifier, name and category of the liked item, and the timestamp of the *like* creation, such as {id: "35481394342", name: "The Matrix", category: "Movie", created_time: "2015-05-14T12:35:08+0000"}. Distinct names may exist for the same item, e.g., "The Matrix", "The Matrix:Film series" and "The Matrix (saga)" for "The Matrix" movie saga. Users thus may provide likes for different Facebook pages referring to the same item. Consolidating and unifying the items of the extracted Facebook *likes*, our method automatically maps the items names to the unique URIs of the corresponding DBpedia entities, e.g., `http://dbpedia.org/resource/The_Matrix` for the identified names of "The Matrix" movie saga. A core stage in the method is to execute SPARQL queries to the DBpedia endpoint that (i) map item names with entity labels, expressed through the `rdfs:label` property, (ii) disambiguate entities using the `rdf:type` property and the Facebook item category field, and (iii) consider equivalent item names by means of the `dbo:wikiPageRedirects` property.

6.3.1 Linking Items to DBpedia Entities

Within the Semantic Web initiative, the Linked Open Data (LOD) project leads the extension of the Web with a global data space connecting diverse semantic entities, such as famous people, organizations, books, movies, music compositions, and reviews, to name a few. Moreover, the consolidation of specialized data storage and information retrieval technologies e.g., the SPARQL[2] RDF query language and the Apache Fuseki[3] server allows accessing LOD similarly to how a relational database is queried today. Among the datasets existing in the Linked Data cloud, DBpedia plays the role of a knowledge hub thus connecting many other data repositories. It is the LOD version of Wikipedia and, as of March 2016, its knowledge base describes 4.58M things, including 1,4M people, 735K places, 411K creative works, and 241K organizations. For each of these things, DBpedia gathers metadata obtained from structured data of the corresponding Wikipedia webpage. Such metadata are represented as triples of the form [*subject* $\rightarrow$ *property* $\rightarrow$ *object*], e.g., the [dbr:The_Matrix, dbo:director, dbr:The_Wachowskis] triple represents that The Matrix movie was directed by the Wachowskis brothers, where dbr: and dbo: are respectively the abbreviations of http://dbpedia.org/resource/ and http://dbpedia.org/ ontology namespaces. As mentioned before, in this work we linked liked items in Facebook with their corresponding DBpedia entities, in order to obtain item metadata with which we can investigate semantic-based collaborative filtering approaches on positively-only feedback. This was done as follows. Given a particular item, we first identified the DBpedia entities that are labelled with the name of the item. For such purpose, we launched a SPARQL query targeted on the subjects of triples that have rdfs:label as property (where rdfs: stands for the http://www.w3.org/2000/01/rdf-schema# namespace) and the item title as object. The next query is an example for The Matrix 2 title.

[2]http://www.w3.org/TR/rdf-sparql-query

[3]http://jena.apache.org/documentation/fuseki2

```
SELECT DISTINCT ?item WHERE {
  {
    ?item rdf:type dbo:Film .
    ?item rdfs:label ?name .
    FILTER regex(?name, "the.*matrix.*2", "i") .
  }
  UNION
  {
    ?item rdf:type dbo:Film .
    ?tmp dbo:wikiPageRedirects ?item .
    ?tmp rdfs:label ?name .
    FILTER regex(?name, "the.*matrix.*2", "i") .
  }
}
```

To resolve ambiguities in those names that correspond to multiple items belonging to different domains, we specify the type of item we wanted to retrieve in each case. Specifically, the query includes a triple clause with rdf:type (or dbo:type) as property, being rdf:the http://www.w3.org/1999/02/22-rdf-syntax-ns#namespace. Hence, in the given example, the subject The Matrix 2 refers to the Movie type, which is associated to the dbo:Film class in DBpedia. The item types were set from the item categories provided in Facebook (see Section 3.1), and their associated DBpedia and YAGO[4] classes were identified by manual inspection of the rdf:type values of several entities. Table 1 shows the list of item types and DBpedia/YAGO classes we considered for the three domains of our dataset. Moreover, running the previous query template we observed that a number of items were not linked to DBpedia entities because the labels corresponded to Wikipedia redirection webpages. In these cases, to reach the appropriate entities the query makes use of the dbo:wikiPageRedirects property. The result of the above query for The Matrix 2 name is: http://dbpedia.org/resource/The_Matrix_Reloaded

[4]http://www.mpi-inf.mpg.de/yago-naga/yago

which actually is the DBpedia entity of the second movie in The Matrix saga. Here, it is important to note that thanks to the Wikipedia page redirect component we are able to link items whose names do not have a direct syntactic match with the label of its DBpedia entity, but with the label of a redirected entity, e.g., the Matrix 2 title matches with The Matrix Reloaded entity.

6.3.2 Final Semantically Annotated Dataset

For every linked entity, we finally accessed DBpedia to retrieve the entity metadata that afterwards would be used as input for the recommendation models. In this case, we launched a SPARQL query asking for all the properties and objects of the triples that have the target entity as subject. Following the example given before, such a query would be:

```
SELECT ?p ?o WHERE {
  dbr:The_Matrix_Reloaded ?p ?o .
}
```

This query returns all the DBpedia property-value pairs of the dbr:The_Matrix_Reloaded entity. However, since our ultimate goal is item recommendation, we should only exploit metadata that may be relevant to relate common preferences of different users. Thus, we filtered the query results by considering certain properties in each domain. Specifically, Table 2 shows the list of DBpedia properties selected for each of the three domains of our dataset. Hence, for example, for the movie items, we would have as metadata the movies genres, directors, and actors, among others. The items and relations shown in the table thus represent a semantic network that is automatically obtained from DBpedia for each particular domain.Table 3 shows statistics of the dataset for the three domains of interest, namely books, movies, and music. The statistics are focused on the number of users, items and ratings from the positive-only feedback side, and the number of properties and triples from the item metadata side.

6.3.3 Semantically Enriched Item Profiles

Fixing books, movies, and music artists and bands as the target items to be recommended, we can distinguish between three types of item metadata. First, the reminder items appearing in the extracted DBpedia semantic networks, and shown in Table 2, can be considered as item attributes, e.g., the genre(s), director(s) and actors of a particular movie. Second, the item-item properties shown in Table 2 derive related items, e.g., the novel that a movie is based on (dbo:basedOn property), the prequel/sequel of a movie (dbo:previousWork / dbo:subsequentWork properties), and the musicians of a band (dbo:bandMember property). Finally, attribute-attribute properties generate **extended item attributes** that originally do not appear as metadata of the items, e.g., the subgenres of a particular music genre (dbo:musicSubgenre property). The above three types of item metadata constitute the semantically enriched item profiles that we propose to use in the recommendation models. We note that they differ from the commonly used content-based item pWrofiles composed of (plain) attributes. We also note that in the conducted experiments, the results achieved by exploiting the enriched profiles were better that those achieved by only using item attributes.

6.4 Experimental setting

Evaluated recommendation methods. We evaluated the following recommendation algorithms in single and cross-domain scenarios, using the validation set to tune model hyperparameters in each case.

POP: Recommends the most popular items not yet liked by the user. **UNN**: User-based nearest neighbors with Jaccard similarity and neighborhood size of $k = 100$. **INN**: Item-based nearest neighbors with Jaccard similarity and indefinite neighborhood size. **IMF**: Hu et al.'s matrix factorization method for positive-only feedback [68] with 29 factors for movies and 21 factors for music.

Thanks to the linking of items to entities in the DBpedia knowledge graph, we are able to exploit algorithms that leverage the graph-based nature of the underlying side information. In particular, we built a hybrid graph as proposed in [109] and we used it as input for the following two algorithms. **HeteRec**: Graph-based recommender system proposed in [172], based on a diffusion method of user preferences following different meta-paths. **PathRank**: Personalized PageRank considering the connectivity between users and items along different meta-paths [86].

See Section 2.6 for more information about HeteRec, PathRank, and the hybrid graph building.

For UNN, INN, IMF, HeteRec and PathRank we considered both their application to single-domain scenarios and to cross-domain ones. Hereafter we use the prefix "CD-" to indicate the cross-domain version of the corresponding algorithm.

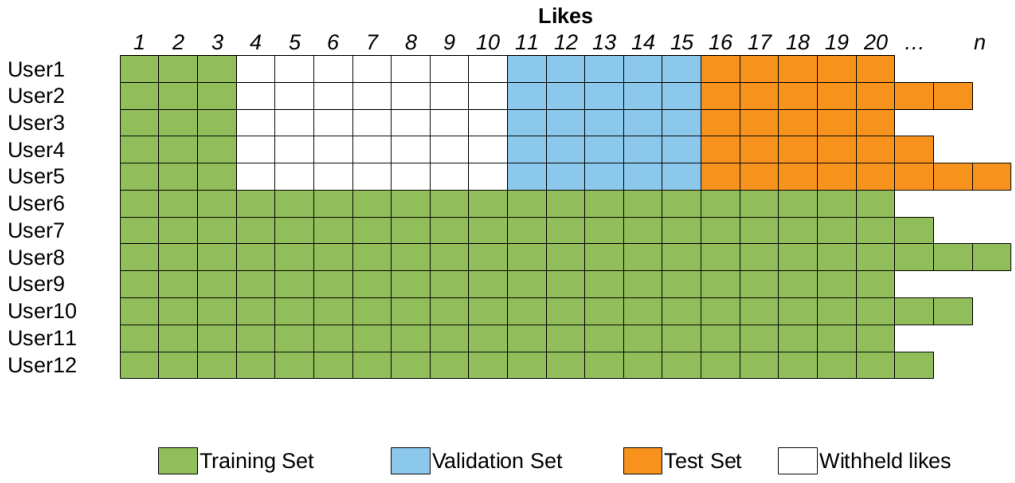


Figure 6.1: Overview of the evaluation strategy for cold-start scenarios [78]. The first five users are the ones in the test set, for whom the profile is resized in order to simulate the cold-start situation (in this case with only three likes in the training set). The other users are used only for the training, therefore their profile is not splitted.

Evaluation methodology. For the evaluation we follow the user-based 5-

fold cross-validation strategy proposed in [78] for cold-start scenarios. First, we select users in the target domain with at least 16 likes and split them into five equally sized subsets. For each fold, we keep all the data from the other folds in the training set, whereas the likes from the users in the selected fold were randomly split into three subsets: training set (10 likes), validation set (5 likes), and testing (remaining likes, hence at least 1). In order to simulate different user profile sizes from 0 to 10 likes, we repeat the training and the evaluation eleven times, starting with the zero like in the training set and incrementally increasing it one by one. This setting allows us to evaluate each profile size with the same test set, avoiding potential biases in the evaluation due to different test set sizes [78]. After this preprocessing, the Facebook music dataset contains 49,369 users, 5,748 music bands or artists, and 2,084,462 likes; the movie dataset contains 26,943 users, 3,901 movies, and 876,501 likes. The user overlap for movies is 89.96% and music is 84.69%. In order to simulate the cross-domain scenario, we simply append the full source domain dataset to the previous training set. We used the Mean Reciprocal Rank (MRR) to evaluate the ranking accuracy of the recommendations, which computes the average reciprocal rank of the first relevant item in the recommendation list. Whereas, Intra-List Diversity (ILD) and Binomial Diversity Framework (BinomDiv) [157] were used to evaluate the individual diversity, namely the degree of diversity in the recommendation lists based on item genres extracted from DBpedia. Along with accuracy, we also measured catalog coverage as the percentage of items that are recommended at least once, to better understand the differences among the compared algorithms.

6.5 Results

In the following we discuss the outcomes of three experiments we conducted to investigate each of the research questions stated in Section 6.1.

	Source – Target	Music – Movies										
	Target size	0	1	2	3	4	5	6	7	8	9	10
MRR	POP	.290	.293	.295	.298	.299	.303	.304	.307	.310	.312	.315
	UNN		334	.324	.322	.330	.345	.379	.393	.402	.412	.422
	CD-UNN	**.383**	279	.304	.321	.335	.347	.353	.368	.378	.394	.406
	INN		233	.308	.334	.359	.374	.388	.403	.408	.420	.426
	CD-INN	.347	**.352**	**.358**	367	.369	.374	.382	.388	.392	.397	.403
	IMF		254	.292	.315	.335	.343	.363	.377	.389	.397	.417
	CD-IMF	.304	.330	.354	**.370**	**.378**	**.387**	**.400**	**.410**	**.424**	**.428**	**.439**
	HeteRec		320	.351	.360	.371	.376	.386	.389	.396	.402	.408
	CD-HeteRec	.376	.345	.350	.356	.361	.364	.367	.370	.374	.381	.384
	PathRank		340	.345	.346	.352	.350	.354	.357	.361	.363	.367
	CD-PathRank	.346	.317	.317	.321	.325	.327	.330	.333	.337	.341	.345

Table 6.1: Accuracy values for different cold-start target profile sizes.

	Source – Target	Movies – Music										
	Target size	0	1	2	3	4	5	6	7	8	9	10
MRR	POP	335	.337	.340	.343	.345	.347	.350	.352	.354	.357	.359
	UNN		.425	.394	.398	.422	.454	.485	.504	.525	**.536**	.547
	CD-UNN	433	.270	.307	.336	.373	.402	.438	.463	.490	.509	.526
	INN		.320	.389	.430	.455	.476	.491	.506	.520	.533	.544
	CD-INN	419	**.437**	**.457**	**.471**	**.480**	**.492**	**.503**	**.514**	**.526**	**.536**	.545
	IMF		.350	.396	.431	.452	.473	.489	.505	.522	.533	**.548**
	CD-IMF	299	.358	.401	.429	.453	.477	.487	.501	.521	.531	.543
	HeteRec		.361	.394	.424	.442	.467	.484	.499	.517	.526	.536
	CD-HeteRec	**.527**	.431	.450	.461	.469	.477	.481	.488	.497	.503	.509
	PathRank		.411	.416	.420	.426	.429	.433	.436	.442	.444	.449
	CD-PathRank	.495	.399	.402	.405	.411	.415	.419	.422	.427	.432	.436

Table 6.2: Accuracy values for different cold-start target profile sizes.

	Source – Target	Music – Movies										
	Target size	0	1	2	3	4	5	6	7	8	9	10
BinomDiv@10	POP	**.401**	.304	.336	.354	.368	.378	.386	**.393**	**.400**	**.405**	**.410**
	UNN		.360	.385	**.404**	**.392**	**.396**	**.394**	**.393**	.393	.396	.395
	CD-UNN	368	**.404**	**.386**	.376	.373	.372	.372	.374	.374	.377	.380
	INN		.289	.308	.315	.321	.323	.327	.329	.332	.333	.337
	CD-INN	309	.240	.268	.283	.297	.304	.310	.316	.322	.325	.330
	IMF		.299	.320	.335	.344	.347	.355	.358	.363	.366	.368
	CD-IMF	.270	.231	.270	.289	.302	.315	.323	.328	.332	.338	.341
	HeteRec		.311	.328	.334	.337	.341	.343	.346	.348	.350	.354
	CD-HeteRec	.333	.271	.298	.314	.324	.333	.339	.345	.350	.354	.358
	PathRank		.317	.327	.336	.342	.352	.353	.359	.361	.366	.368
	CD-PathRank	.336	.270	.294	.310	.320	.327	.334	.339	.345	.350	.355

Table 6.3: Diversity values for different cold-start target profile sizes.

	Source – Target	Movies – Music										
	Target size	0	1	2	3	4	5	6	7	8	9	10
BinomDiv@10	POP	.324	.228	.262	.282	.295	.305	.313	.321	.327	.333	.338
	UNN		.296	.332	.348	.347	.330	.317	.309	.305	.301	.297
	CD-UNN	.296	**.411**	**.380**	.358	.347	.329	.322	.316	.311	.308	.305
	INN		.200	.213	.219	.223	.229	.231	.235	.236	.239	.240
	CD-INN	.277	.231	.255	.264	.270	.272	.273	.274	.274	.276	.276
	IMF		.196	.217	.232	.241	.249	.253	.256	.260	.261	.265
	CD-IMF	.248	.229	.254	.264	.271	.272	.276	.277	.277	.278	.278
	HeteRec		.227	.264	.280	.288	.296	.300	.302	.304	.306	.306
	CD-HeteRec	.372	.271	.314	.331	.342	.349	.354	.357	.361	.363	.366
	PathRank		.350	**.380**	**.395**	**.404**	**.410**	**.413**	**.416**	**.419**	**.421**	**.422**
	CD-PathRank	**.405**	.335	.367	.384	.394	.402	.408	.412	.415	.418	.419

Table 6.4: Diversity values for different cold-start target profile sizes.

Cross-domain recommendation accuracy To address RQ1, we compare the accuracy of the target recommendations in single-domain and cross-domain scenarios. Tables 6.1 and 6.2 show the MRR values for movies and music target recommendations, respectively.

Music (source)–Movies (target). CD-UNN is the most accurate method for extreme cold-start users (0 likes in target domain), CD-INN where 1 or 2 likes are provided, and CD-IMF from 3 to 10 likes. Curiously, CD-INN and CD-HeteRec using only cross-domain information are able to beat almost all the other methods even where they use target information up to 4 likes. Moreover, CD-UNN is subject to a drastic fall from 0 to 1 like, obtaining the worst accuracy among all the methods and configuration (even lower than POP). Further analysis revealed that this is due to our choice of Jaccard as user similarity metric, which we observed provides unreliable scores in cold-start situations. Comparing the methods between single and cross-domain configuration, we can see that only INN and IMF can benefit from music feedback in terms of accuracy. All the other methods lose accuracy when music feedback is also considered. In terms of coverage, UNN is the only method able to benefit from music feedback: UNN reaches values from 10% to 18% and CD-UNN from 38% to 50% among the different profile sizes.

Movies–Music. CD-HeteRec yields the most accurate recommendations in the extreme cold-start scenario, while CD-INN is the best method for all the other profile sizes, even though UNN obtains close values with 8 and 9 likes, and UNN and IMF overcome CD-INN with 10 likes but with a not relevant difference. CD-UNN shows again a drastic loss from 0–4 target likes, falling even below POP. In terms of catalog coverage, the trends are very similar to the ones in movies domain. Interestingly, CD-INN beats again all the other methods in terms of accuracy and catalog coverage with 1 and 2 likes in the target domain. Analyzing the use of cross domain information, INN is once again able to reach better accuracy using the additional movie likes, while HeteRec obtains a benefit where less feedback is provided (from

1 to 5). Again, CD-HeteRec with 0 likes in the target domain overcomes all the other methods even where they use more target information (up to 8). However its catalog coverage is too low (1%) compared the other methods (>10%).

Summing up, we may say that cross-domain information is arguably useful to face the cold-start user problem, allowing to generate relevant recommendation even where no target information is available. The choice of the method depends on the domain and amount of user information available. Moreover, we discover that some methods obtain exceptionally better results using only the source domain rather than using a few target feedbacks as well. More research will be needed for better understanding this trend.

Cross-domain recommendation diversity This section addresses RQ2, namely testing whether cross-domain information leads to more diverse recommendations. Tables 6.3 and 6.4 shows the diversity results for movies and music, respectively, as target domains in terms of BinomDiv@10. We also compared the methods using the ILD metric, but we do not show its values, since they obtain a very similar trend to BinomDiv.

Music–Movies. POP obtains good results values, since all the most popular movies in the dataset belong to different genres, but CD-UNN and UNN overcome it with 1 and 2 likes, and only UNN from 3 to 6. In general, using cross-domain music information yields to less diverse recommendations.

Movies–Music. PathRank and CD-PathRank produce the most diverse recommendations. Conversely, MF methods lead to the worst diversity. In contrast to the previous situation, using cross-domain movies information for music recommendations improves nearly always the diversity degree of the recommendations.

Size and diversity of source domain user profiles In order to address RQ3, we compute the number of preferences and the intra-list diversity of the user profiles in the source domain, and group users in different ranges. For the profile sizes we split users in intervals of 20 likes, from size 0 to

100 and beyond, and for profile diversity we classify the users in terms of the distribution of ILD scores. Specifically, we define four groups based on the quartiles which we name `Very low` (0–25%), `Low` (25%–50%), `Medium` (50%–75%), and `High` (75%–100%). Finally, we average the MRR of the recommendation lists in the target domain separately for each group, first in terms of profile size and then in terms of diversity. Figure 6.2 shows the relation between the quality of the target recommendations and the analysed source profile properties. We only report the results for the extreme cold-start profile sizes in the target, i.e., 0 and 10, as the rest showed similar behavior.

In terms of source profile size, we notice that in general the quality of target recommendations improves monotonically as more information about the user's preferences is available in the source domain. This trend holds for all the evaluated algorithms with the exception of CD-IMF in music, where we see that the performance degrades when the size of the source profile is larger than 100. In this case, we argue that the abundance of auxiliary preferences could be drifting the learning of the model parameters towards the source domain, although a deeper analysis is needed to confirm our intuition.

Regarding the impact of the source profile diversity we find that the best results are achieved for users very focused on limited types of items, whereas a more diverse profile has a negative effect on the accuracy of the recommendations. This seems to indicate that the evaluated algorithms struggle to find inter-domain correlations, specially from music to movies. In the case of very high diversity we see that the two settings diverge: variety in source movie preferences is beneficial for music recommendations, whereas the converse has the opposite effect.

We conclude that both the source user profile size and diversity have a significant impact on the quality of cross domain recommendations, thus confirming RQ3. On a side note, we observe the superior performance of CD-INN in most of the considered scenarios, specially in the extreme cold-start

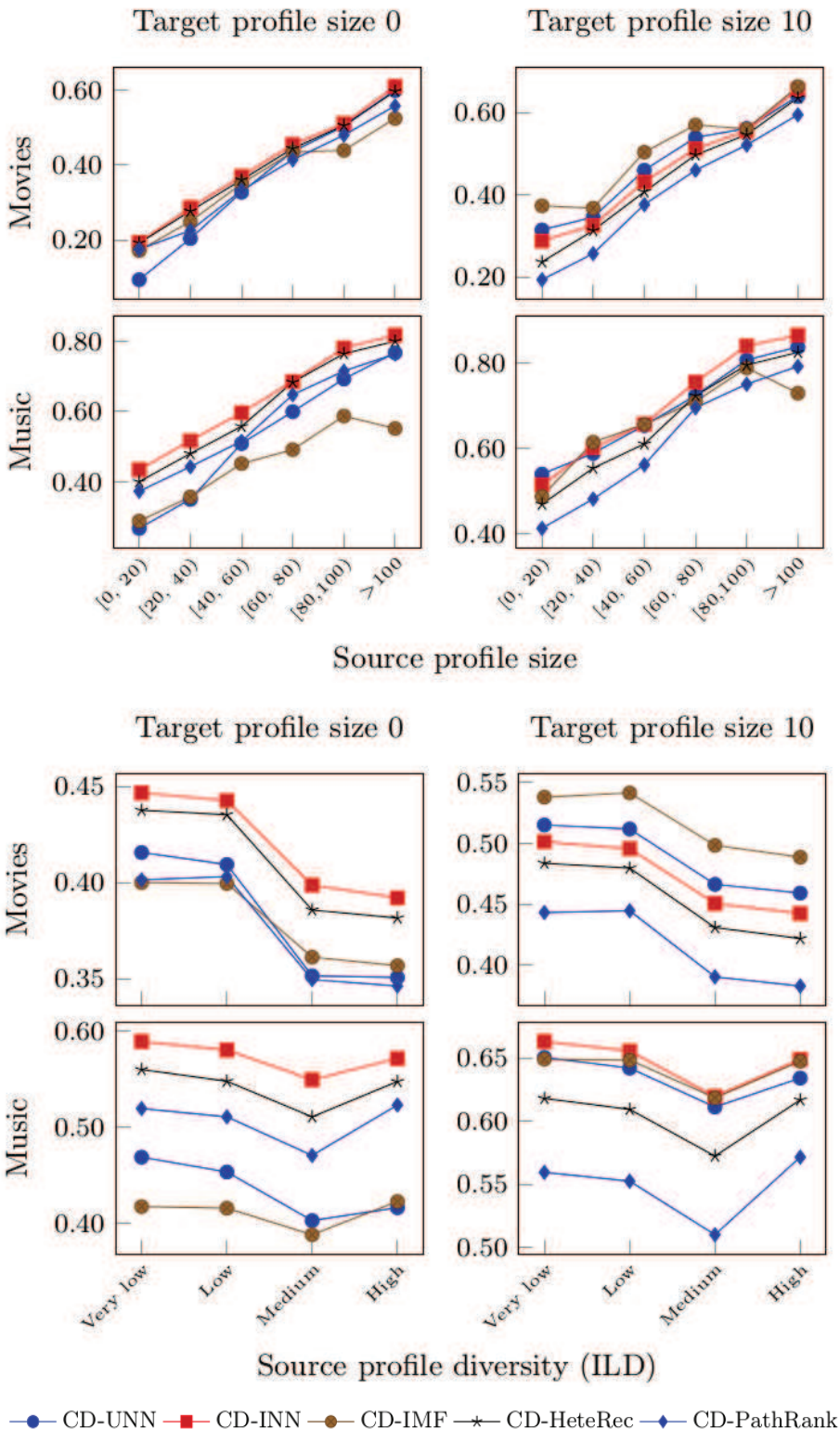


Figure 6.2: MRR values of the cross-domain recommendation methods for different user profile sizes and profile diversity values in the source domain. Each row corresponds to the two target domains in our dataset.

with target profile size of 0. We argue that this behavior is a consequence of the relatively large overlap of users between the analysed domains, an issue that we plan to further investigate in future work.

6.6 Summary

We have studied the quality of cross-domain recommendations in terms of accuracy, diversity and catalog coverage, evaluating a number of algorithms on two datasets with positive-only feedback. Our results show the benefits of cross-domain information in cold-start situations in terms of ranking accuracy. Regarding diversity we observe different behavior in the two datasets, and therefore conclude that in general the results depend on the target domain. We have also studied the impact of the size and diversity of user profiles in the source domain, concluding that while more cross-domain user preferences are helpful, a greater item diversity in the source profile can actually harm the performance in the target domain. Following this work we intend to further investigate which characteristics of the datasets could explain the differences we found in both recommendation and user profile diversity. We plan to extend our analysis to more domains, e.g. books, and to evaluate more sophisticated methods from the state of the art, such as [66].

Chapter 7

Accuracy and Diversity in Graph-based Similarity Metrics for Content-based Recommender Systems

In this work, we review two existing metrics, SimRank and PageRank, and investigate their suitability and performance for computing similarity between resources in RDF graphs and investigate their usage to feed a content-based recommender system. Finally, we conduct experimental evaluations on a dataset for musical artists and bands recommendations thus comparing our results with two other content-based baselines measuring their performance with precision and recall, catalog coverage, items distribution and novelty metrics.

7.1 Introduction

The Web of Data is the natural evolution of the World Wide Web from a set of interlinked documents to a set of interlinked entities. It is a graph of information resources interconnected by semantic relations, thereby yielding the name Linked Data. The proliferation of Linked Data is for sure an opportunity to create a new family of data-intensive applications such as recommender systems. In particular, since content-based recommender systems base on the notion of similarity between items, the selection of the right graph-based similarity metric is of paramount importance to build an effective recommendation engine. In this section we investigate the effectiveness of SimRank and Personalized PageRank as a means for measuring semantic similarity in RDF graph to build a content-based recommendation engine. SimRank has been employed effectively for measuring similarity in homogeneous graphs [170] and similarly, PageRank has also been successfully exploited in calculating similarity for WordNet [10, 11]. We investigate how effectively the two metrics work with Linked Data.

In particular, we identify two main research questions:

- **RQ1** ***How beneficial in terms of accuracy, novelty and aggregate diversity are content-based graph-based RSs with respect to algorithms based on vector space model?***
- **RQ2** ***What is the best choice between SimRank and Personalized PageRank?***

To answer at the aforementioned questions, we perform experiments on data retrieved from DBpedia[1], the cornerstone of the whole Linked Open Data cloud[2], to feed a content-based recommender system (RS) with the aim of evaluating the effectiveness of the two metrics. Indeed, the knowledge encoded in semantic datasets of the Linked Open Data project can be exploited to improve the performance of content-based (CB) approaches

[1] http://dbpedia.org
[2] http://lod-cloud.net/

to recommendation [42]. Such systems try to recommend items similar to those a given user has liked in the past, matching up the attributes of a user profile in which preferences and interests are stored, with the attributes of the items descriptive content [93]. We use the values computed with SimRank and Personalized PageRank, to find similarities between items and use them to produce the final recommendation list. Our experimental evaluation has been conducted by using the well known dataset of Last.fm[3] for musical artists recommendation. We compare our recommendation results with two other content-based baselines measuring their accuracy with precision and recall metrics as well as measuring catalog coverage, items distribution and novelty.

The main contributions of the work are: (i) the evaluation of the effectiveness in the adoption of two well-established graph-based ranking metrics in a *pure content-based RS* scenario; (ii) the analysis of their performances in terms not only of precision and recall but also of diversity, items coverage and novelty.

7.2 Related Work

In this section we briefly recall some related work adopting a content-based (or hybrid) approach to recommendation by exploiting a semantic-based approach. A system for recommending musical artist and bands based on `DBpedia` is presented in [116]. The systems extracts data about bands and artists from `DBpedia`. Instances, namely resources instantiating the classes `dbo:MusicalArtist` and `dbo:Band`[4], are analyzed to supply input data for the recommendation algorithm. A similarity measurement algorithm, called LDSD, is designed to calculate similarity between a piece of music or an artist and the elements of a candidate set. This semantic similarity metric is com-

[3] http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/

[4] In the rest of the chapter, for the sake of clarity and compactness, we will use CURIEs instead of URIs. Moreover, the prefixes for CURIEs are the ones available on http://prefix.cc.

puted on the basis of links between resources. Based on the filtering results, the system produces a list of songs or artists that can then be presented to users. In [143], a framework for evaluating artist similarity is built. Given an artist, the system searches for alike artists according to two factors: style and mood. First, the information for style and mood is collected from the All Music Guide website. Afterwards, a co-clustering algorithm is employed to build hierarchical taxonomies describing the two factors. With respect to the taxonomies, the similarity between each pair of terms is computed by means of the existing similarity metrics: Resnik [124], Jiang-Conrath [73], Lin [89] and Schlicker [140]. From the similarity calculation, a list of similar artists for a selected artist is eventually generated. A key function for content-based recommender systems is calculating similarity between items, so that possible items can be recommended to a user based on his past selected preferences. The authors in [111] exploit `Linked Data` to compute similarity between resources for a content-based recommender system. They propose a neighborhood-based graph kernel to compute semantic similarity. In this approach, items are represented as nodes in a neighborhood graph. Starting from two resources, a graph is extended by using a set of selected properties. Each node involved in the similarity calculation is assigned a weight corresponding to its relationship with other neighborhood nodes. Afterwards, a kernel function is devised to calculate similarity between them. The effectiveness of the approach has been proven by the experimental results on the Movielens dataset. A content-based system leveraging `DBpedia` for recommending movies is introduced in [42]. Based on the set of movies that have been preferred by a user, the system engine extracts movie information from `DBpedia` and computes the similarity between a new item and current rated items. A movie is characterized by a set of features corresponding to the neighbor movies in the graph via a property. The set of features is represented as a vector. The cosine function is used to measure semantic similarity. By calculating similarity between two movies, the system can provide a user with a list of similar movies according to his previously selected prefer-

ences. For computing *top-N* item recommendations from implicit feedback, a hybrid algorithm - named SPrank - is proposed in [110]. The algorithm extracts path-based features from `DBpedia` and mines semantic graph to detect subtle relationships among items. It incorporates ontological knowledge with collaborative user preferences to produce recommendations. Experimental results on two datasets from MovieLens and Last.fm demonstrate that the proposed algorithm gains a high prediction accuracy even when the experimental data is sparse. In [61], a hybrid recommender system has been proposed to overcome the problems of cold-start and lower accuracy in collaborative and content-based recommender systems. The proposed approach models item interactions using unified Boltzmann machines. By integrating collaborative and content information, the system learns weights representing the importance of different pairwise interactions. Based on the probabilistic models, the system can predict whether a user will act on a specific item. By doing that, more appropriate recommendations can be made.

7.3 Graph-based Semantic Similarity Metrics

An `RDF` graph is defined as a directed graph `G=(V,E,R)`, where `V` is the set of vertices, `E` is the set of edges and `R` represents the relationship among the nodes. An `RDF` graph consists of enormous nodes and oriented links with semantic relationships. Its building block is the triple <`subject, predicate, object`> stating that the node `subject` is connected to the node `object` by means of the edge labelled with `predicate`. To evaluate how similar two given resources within an `RDF` graph are, it is necessary to incorporate their intrinsic characteristics into the similarity calculation. To be precise, nodes, links, and the mutual relationships among subjects and objects could be considered as input for the calculation. Although originally developed for homogeneous graphs, Personalized PageRank and SimRank can be two interesting candidates to compute similarity between `RDF` resources. For the sake of completeness, in the following we briefly recall the way they are com-

puted.

SimRank has been proposed to compute similarity between nodes in a graph using the structural context [71]. Similarity is calculated according to object-to-object relationships: the similarity between two nodes is dependent on their neighbors. Two nodes are considered to be similar if they are referenced by similar nodes. The similarity value for two nodes α and β is computed by SimRank using a fixed-point function. Given $k \geq 0$ we have $R^{(k)}(\alpha, \beta) = 1$ with $\alpha = \beta$. Dually, we have $R^{(k)}(\alpha, \beta) = 0$ with $k = 0$ and $\alpha \neq \beta$. In all the other cases the general formula is

$$R^{(k+1)}(\alpha, \beta) = \frac{d}{|I(\alpha)| \cdot |I(\beta)|} \sum_{i=1}^{|I(\alpha)|} \sum_{j=1}^{|I(\beta)|} R^{(k)}(I_i(\alpha), I_j(\beta)) \tag{7.1}$$

where d is a damping factor ($0 \leq d < 1$); $I(\alpha)$ and $I(\beta)$ are the set of inbound neighbors of α and β, respectively. $|I(\alpha)| \cdot |I(\beta)|$ represents a normalization factor to have $R^{(k)}(\alpha, \beta) \in [0, 1]$. Equation (7.1) implies that the similarity for two nodes is computed by aggregating the similarity of all possible pairs of their neighbors. SimRank has been originally designed for homogeneous graphs.

PageRank computes the rank of a node according to the relationship with other nodes. A node receives an amount of rank from every node which points to it and in turn transfers an amount of its rank to the nodes it refers to. In this sense, a node will have a high rank if it is referenced by nodes with high rank. To compute the rank of n nodes, an $n \times n$ transition matrix G is built from the link relationship between the nodes. In this matrix, row i represents the rank that node α_i transfers to other nodes that it has links to. Given the set $O(\alpha)$ representing the set of outbound links of α, it transfers the amount of rank $rank(\alpha) = \frac{1}{|O(\alpha)|}$ to all of its neighborhood nodes. In the transition matrix, the cell at row i and column j has the value of $rank(\alpha)$ if there is a link from node α_i to node α_j, otherwise it has the value of 0. From this definition, a problem arises with *dangling nodes*, i.e. nodes with no outgoing links. By these nodes, the PageRank vectors degrade very quickly and produce inappropriate ranks. To circumvent

dangling nodes, some amendment is made to the transition matrix. This is done by introducing two vectors thus redefining the transition matrix as $G' = G + \vec{\delta} \cdot \vec{\omega}$, where $\|\vec{\omega}\|_1 = 1$ and δ is a column vector with $\delta_i = 1$ if i is a dangling node and $\delta_i = 0$ otherwise. Usually, all entries ω_i are set to $\frac{1}{n}$. Based on the original transition matrix, the Google matrix used for PageRank is defined as follows [165] as $G'' = d \cdot G' + (1 - d) \cdot \vec{v}$ where d is the damping factor ($0 \leq d < 1$) and $\vec{v}$ is the personalization vector. The outcome vector represents the ranks of all nodes, i.e. entry i holds the rank of the graph node α_i. Similar to SimRank, the PageRank vector is obtained after a finite number of iterations. The complete formulation is:

$$\vec{\pi}^{(k+1)} = d \cdot \vec{\pi}^{(k)} \cdot G + d \cdot (\vec{\pi}^{(k)} \cdot \vec{\delta}) \cdot \vec{\omega} + (1 - d) \cdot \vec{v} \tag{7.2}$$

A variant of the original PageRank algorithm, the Personalized PageRank algorithm was derived to measure the similarity between topics [63]. The main idea of this approach is to exploit PageRank vector to characterize a topic which is comprised of a set of words. The topic is characterized by concentrating probability mass to its constituent words, represented as nodes in a graph. This is done by modifying the personalization vector $\vec{v}$ in Equation (7.2). The corresponding entries in $\vec{v}$ are assigned the value of 1, whilst all the other entries of $\vec{v}$ are assigned the value of 0. By doing this the biased topic will earn a high rank. The PageRank vector obtained is considered as the features of the topic and helps distinguish the topic from others. The similarity between two topics α and β represented by vectors $\alpha = \{a_i\}_{i=1,..,n}$ and $\beta = \{b_i\}_{i=1,..,n}$ is computed as the inner product space between the two vectors [155].

$$p.PageRank(\alpha, \beta) = \frac{\sum_{i=1}^{n} a_i \times b_i}{\sqrt{\sum_{i=1}^{n} (a_i)^2} \times \sqrt{\sum_{i=1}^{n} (b_i)^2}}$$

Personalized PageRank has been applied to measure similarity between words in WordNet [10, 11]. Following the same line of reasoning, we believe that Personalized PageRank can be used for computing similarity between resources in an `RDF` graph. In the preceding sections, we are going to em-

ploy SimRank and Personalized PageRank to measure similarity between resources in Linked Data.

7.4 Experimental setting

A question that might arise at any time is how effective the two metrics are with regards to a recommendation task? In this section we present our attempt to analyze the performance of SimRank and PageRank. Such a study is of particular importance since it not only provides an insight into the suitability but also casts light on the effectiveness of measurement techniques for Linked Data. In our experiments, we re-implemented the two metrics, SimRank and PageRank and we conducted experiments on RDF graphs. In particular, as input for the calculation, we retrieved data from DBpedia via SPARQL queries and extracted a subgraph containing only the information related to a specific domain. In particular, we considered data from the music domain. We retrieved resources that are instances of the two classes dbo:MusicalArtist and dbo:Band. For each resource we also got the RDF triples that are involved in as subject or object. The final outcome is then used in a content-based recommendation engine to evaluate its performance in terms of accuracy, novelty, items distribution and sales diversity. The latter is considered a relevant quality dimension for both business and user perspective: the user may receive less obvious recommendations, comply with the objective to help users discover new content [160] and the business may increase the sales [53].

From a computational point of view, evaluating both SimRank and Personalized PageRank values is a high demanding task that strongly depends on the number of edges connecting nodes within the graph. In order to reduce the computational load we downsized the extracted subgraph by selecting a set of RDF properties that result meaningful for the domain of interest (music in our case). We selected those properties belonging to the DBpedia ontology (plus dcterms:subject) that occur most frequently in the dataset in rela-

tion to musical artists and bands. In particular we selected incoming and outgoing properties as shown in Table 7.1.

Inbound	*Outbound*
dbo:producer	dcterms:subject
dbo:artist	dbo:genre
dbo:writer	dbo:associatedBand
dbo:associatedBand	dbo:associatedMusicalArtist
dbo:associatedMusicalArtist	dbo:instrument
dbo:musicalArtist	dbo:occupation
dbo:musicComposer	dbo:birthPlace
dbo:bandMember	dbo:background
dbo:formerBandMember	
dbo:starring	
dbo:composer	

Table 7.1: **The set of properties used to compute similarity values between pairs of resources.**

	Last.fm
Number of users	1,867
Number of items	700
Number of ratings	47,330
Data sparsity	0.963%
Avg number of users per item	98.19
Avg number of items per user	25.52
Number of extracted triples	113,386
Avg number of Inbound links	109.25
Avg number of Outbound links	31.44

Table 7.2: **Statistics about the Last.fm dataset.**

In order to show the quality of similarities computed with SimRank and PageRank for a content-based recommender system, we have carried out experiments to evaluate the proposed recommendations. We selected the well known dataset `Last.fm hetrec-2011`[5].

[5]Available at http://ir.ii.uam.es/hetrec2011/datasets.html

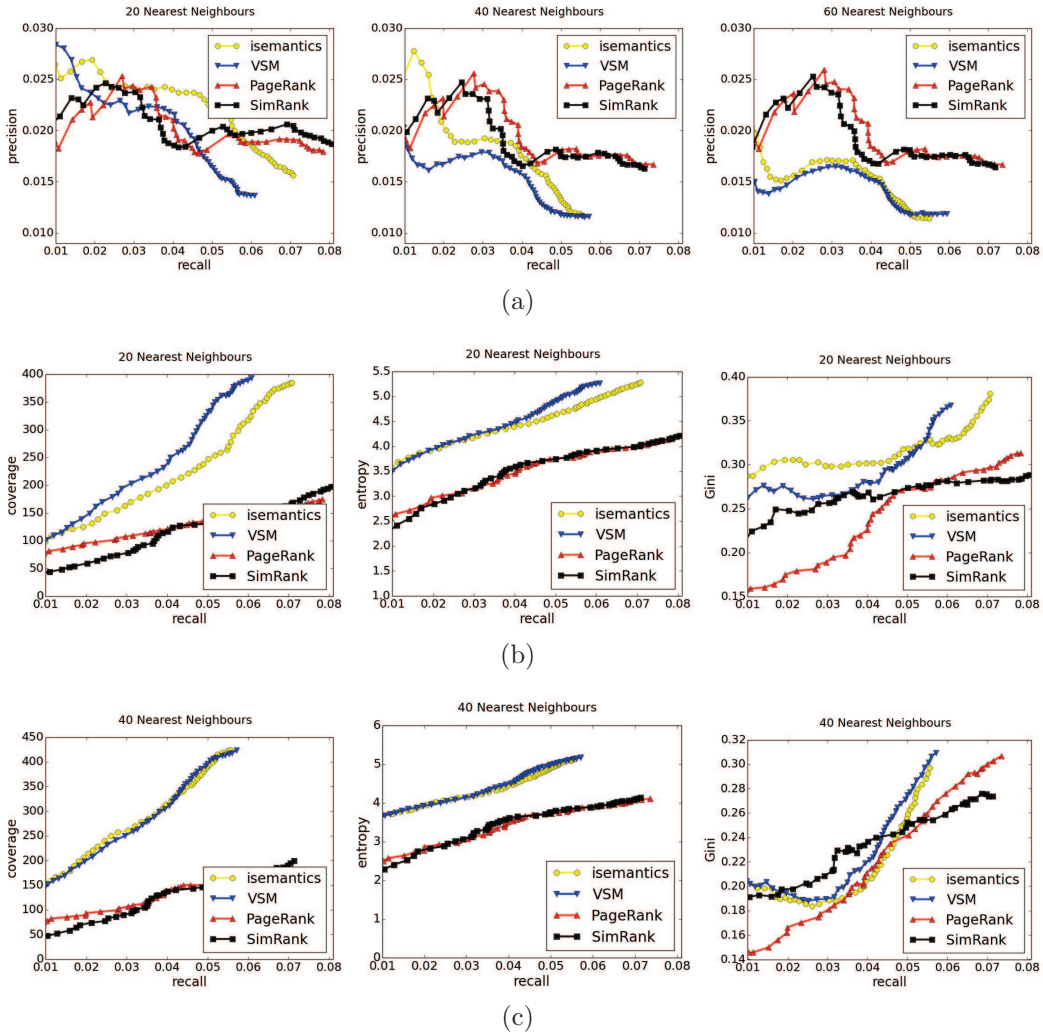


Figure 7.1: (a) Precision and Recall curves obtained by varying the length of the recommendations list from 1 to 50 on the Last.fm dataset. Catalog coverage, Entropy and Gini index curves obtained by varying the length of the recommendations list from 1 to 50 on Last.fm, with 20 (b) and 40 (c) neighbors. The results with 60 neighbors are very similar to those with 40 neighbors, therefore they are omitted due to space limitations.

Last.fm contains 92835 implicit feedbacks on 17632 artists by 1892 users. For computational reasons we downsized the number of artists and bands to the 700 most popular ones. Then a mapping of each item to the corre-

sponding DBpedia URI has been used by exploiting the data available at http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/. For the music domain we then extracted the corresponding subgraph composed by a total of 113, 386 triples. In Table 7.2 we summarize some statistics of the data we used. We used for our experiment the k-nearest neighbors algorithm, which finds the set $neighbors(\alpha)$ containing the k most similar entities β to a given item α using a similarity function $sim(\alpha, \beta)$ [93]. For our experiments, we have implemented a content-based recommender system using a k-nearest neighbors algorithm (see Section 2.2 and Formula 2.3). Given an item-based similarity measure, INN approach estimates the rating of user u for item i using the following formula:

$$r^*(u,i) = \frac{\sum_{j \in N(i) \cap r(u)} sim(i,j) \cdot r(u,j)}{\sum_{j \in N(i) \cap r(u)} sim(i,j)}$$

where $r(u)$ represents the items rated by the user u, and $r(u, j)$ the rating value given by the user u with respect to the item i. The experiments have been carried out by considering as $sim(i, j)$ the values computed via Equation (7.1) for SimRank and Equation (7.3) for Personalized PageRank. In order to compare our results, we also selected for our experiments two more similarity functions as baselines: a pure vector space model with cosine similarity (VSM) and a simplified variation of the algorithm proposed in [42] (isemantics).

7.5 Results Discussion

In order to estimate the quality of the recommendations and respond the research questions described in the introduction, we used the holdout method splitting a dataset into two parts: training set and test set. We built the training set by using, for each user, the first 60% of the ratings and the remaining 40% to build the test set. Last.fm has implicit feedback, therefore each item in the test set was perceived as relevant. For evaluating recommendation ranking accuracy we used the *TestItems* evaluation methodology

presented in [18]. Considering only the top N results, for measuring accuracy we used precision (P@N) and recall (R@N). The former is defined as the fraction of the top-N recommended items that are relevant to the user u; the latter as the fraction of relevant items from the test set that appear in the N predicted items.

To measure the sales diversity, we considered three other important metrics: catalog coverage [58] (the percentage of items in the catalog that are ever recommended to users), Entropy and Gini coefficient to measure the distribution of recommended items [6, 53, 160] (the degree to which recommendations are concentrated on a few items or are more equally distributed across the items). As in [6], we reversed the scale for Gini coefficient so that smaller values represent lower distributional equity and larger values correspond to higher equity.

Furthermore, we used two metrics to measure the novelty of the recommendations: long-tail percentage [6] and Expected Popularity Complement (EPC@N) [160]. The former measures the percentage of long-tail items among the recommendations across all users. The Expected Popularity Complement metric [160] measures the novelty of a recommendation list as the average novelty of the recommended items L_u. More formally, $EPC(L_u) = \frac{\sum_{\alpha \in L_u} nov(\alpha)}{|L_u|}$ where $nov(\alpha)$ measures the probability of not to being known by a random user $nov(\alpha) = 1 - \frac{\sum_{u \in U} 1_{r(u,\alpha)>0}}{\sum_{u \in U} |profile(u)|}$, being U the set of all the users.

The results for the four algorithms were computed with 10, 20, 40, 60, 80 neighbors for different number of top-N recommendations (from top-*1* to top-*50*). Due to space limitations, only the results with 20, 40, 60 neighbors are shown in Figure 8.1[6]. Figure 7.1(a) shows the results on Last.fm in terms of precision and recall. SimRank and Personalized PageRank can produce comparable results and they outperform the two baselines when the number of selected neighbors increases. This means that they are not affected by possible noise if more neighbors are considered. Moreover, we

[6]Results with 10 and 80 neighbors are very similar to the ones with 20 and 60 neighbors respectively.

	Top10		Top20		Top30		Top40		Top50	
	EPC	Long-tail%	EPC	Long-tail%	EPC	Long-tail%	EPC	Long-tail%	EPC	Long-tail%
isemantics	.869	.695	.844	.648	.841	.657	.839	.658	.836	.643
VSM	.865	.677	.848	.657	.847	.675	.848	.673	.843	.658
PageRank	**.913**	**.780**	**.913**	**.823**	**.893**	**.767**	.867	**.743**	.860	.710
SimRank	.911	.796	.906	.793	.881	.747	**.874**	.742	**.867**	**.717**

Table 7.3: **Comparison of novelty results in terms of EPC@N and Long-tail%@N for the four algorithms with 60 neighbors.**

see that starting from 40 neighbors, their results tend to stabilize. They obtain higher recall values in each configuration, thus strongly surpassing the baselines with the higher top-N (particularly over top-30). This means that by increasing the number of recommendations, the two metrics are able to suggest other relevant items, in this sense they perform better than the two baselines. We also observe that they obtain lower values for catalog coverage and dispersion compared to the two baseline. However in Table 7.3 we see that they outperform the baseline in terms of EPC and long-tail percentage. This means that they tend to suggest always a small subset of items and the suggestions are not equally distributed, but these items do not necessarily belong to the most popular ones. As an answer to the question RQ1, the two graph-based methods analysed in the this work – i.e. SimRank and Personalized PageRank – provide more accurate and novel recommendations, but penalize the aggregate diversity, in terms of catalog coverage and distribution. Answering to the question RQ2 is very difficult, since the two graph-based methods lead to very similar results.

7.6 Summary

In this section we have presented an investigation on the usage of SimRank and Personalized PageRank for measuring similarity based on the structural context of a graph in order to feed a content-based recommender system. In particular, we analyzed the potential of using these metrics for automatically measuring similarity when the data describing the content are available as

`Linked Data`. To validate the outcomes, we conduct experimental evaluations on the Last.fm dataset thus comparing our recommendation results with some other standard content-based baseline measuring their accuracy with precision and recall metrics. Moreover we measured the performance of the recommender system in terms of catalog coverage, items distribution and novelty of results. As an answer to the question RQ1, experimental results show that, in the given circumstances, SimRank and Personalized PageRank can produce interesting results compared to the two baselines in terms of precision, recall an novelty even though their performance decrease when we evaluate catalog coverage, items distribution. However, the two graph-based methods obtain very similar results and hence we cannot respond to the question RQ2. Further research is needed to assess potential differences between them, that may perhaps depend on the domain and dataset dimension.

Chapter 8

Accuracy and Diversity in Content-based recommendations via DBpedia and Freebase: a comparative analysis

The recent boom in Linked Data facilitates a new stream of data-intensive applications that leverage the knowledge available in semantic datasets such as DBpedia and Freebase. These latter are well known encyclopedic collections of data that can be used to feed a content-based recommender system. In this chapter we investigate how the choice of one of the two datasets may influence the performance of a recommendation engine not only in terms of precision of the results but also in terms of their diversity and novelty. In particular, we tested four different recommendation approaches exploiting

both DBpedia and Freebase in the music domain.

8.1 Introduction

The Linked Open Data cloud has been launched in an effort to transform structured data into first class citizens in the Web thus moving it towards the so called Web of Data. The data published as Linked Data (LD) by means of RDF covers a wide range of knowledge, including life science, environment, industry, entertainment, to name a few. The new data platform paves the way for several fresh applications but the proliferation of LD is overshadowed by the fact that the quality of the newly uploaded data is yet to be thoroughly verified [79] and that the selection of the dataset may heavily influence the performance of an LD-based tool. Among all possible data intensive applications, recommender systems are gaining momentum to potentially profiting from the knowledge encoded in LD datasets. As background data is of crucial importance to recommender systems, one should consider the suitability of a dataset when designing a recommender system since it may depend on the type of tasks as well as the recommendation algorithm. A reasonable combination of the underlying data and recommendation approach might contribute towards a great difference in performance. This motivates us to perform an investigation on the adequacy of a dataset when adopting a recommendation strategy.

In this chapter we evaluate the fitness for use of LD sources to feed a pure content-based recommender system [93] and in particular we examine the suitability of two encyclopedic data sources namely DBpedia[1] and Freebase[2] for musical artists recommendation tasks. As the input for the calculation we exploit similarity values computed by four different feature-based semantic similarity metrics. The values are used to find similarities between items and eventually to produce the final recommendation list.

[1] http://dbpedia.org

[2] http://www.freebase.com/

In particular, we identify two main research questions:

- **RQ1** ***What is the best choice between DBpedia and Freebase as knowledge bases for feeding CBRSs in terms of different recommendation quality dimensions?***
- **RQ2** ***What is the best choice among the four similarity metrics for DBpedia and Freebase?***
- **RQ3** ***How beneficial is moving from one to two hops in the knowledge graph in terms of different recommendation quality dimensions?***

Our experimental evaluations are conducted by using the well-known dataset Last.fm for musical artists recommendation[3]. To study the fitness for use of the data sources to recommendation tasks, we conducted an offline evaluation and we analyzed three different dimensions: *Accuracy*, *Aggregate Diversity*, and *Novelty*. Various indicators are employed to analyze the recommendations pertaining to these characteristics. The main contributions of the chapters can be summarized as follows: (a) evaluating the fitness for use of `DBpedia` and `Freebase` as input for content-based recommendation tasks in the music domain by means of various quality dimensions and quality indicators; (b) providing an evaluation of the performance for four semantic similarity metrics, with regard to recommendation tasks, on the aforementioned encyclopedic datasets.

8.2 Related Work

To the best of our knowledge, none of the existing work has conducted a comprehensive evaluation on the fitness for use of datasets in combination with different recommendation strategies. Some studies partly address the issue in different settings. In this section we review the most notable work on this topic.

[3] http://ir.ii.uam.es/hetrec2011/datasets.html

Leveraging LD sources like DBpedia for recommendation tasks appears to be highly beneficial as demonstrated by numerous applications. One of the first approaches that exploits LD for building recommender systems is [65]. The authors of [50] present a knowledge-based framework leveraging DBpedia for computing cross-domain recommendations. A graph-based recommendation approach utilizing model- and memory-based link prediction methods is presented in [92]. LD datasets are exploited in [95] for personalized exploratory search using a spreading activation method for finding semantic relatedness between items belonging to different domains. For recommending movies, a content-based system exploiting data extracted from DBpedia has been proposed in [42] based on the adaptation of Vector Space Model to semantic networks. In [110] a hybrid algorithm - named *Sprank* - is proposed to compute *top-N* item recommendations from implicit feedback. Path-based features are extracted from DBpedia to detect subtle relationships among items in semantic graphs. Afterwards, recommendations are produced by incorporating ontological knowledge with collaborative user preferences. The proposed algorithm gains good accuracy, especially in conditions of higher data sparseness. A full SPARQL-based recommendation engine named RecSPARQL is presented in [14]. The proposed tool extends the syntax and semantics of SPARQL to enable a generic and flexible way for collaborative filtering and content-based recommendations over arbitrary RDF graphs. The authors of [150] propose an approach for topic suggestions based on some proximity measures defined on the top of the DBpedia graph.
In [76] the authors present an event recommendation system based on LD and user diversity. A semantic-aware extension of the SVD++ model, named SemanticSVD++, is presented in [130]. It incorporates semantic categories of items into the model. The model is able also to consider the evolution over time of user's preferences. In [131] the authors improve their previous work for dealing with cold-start items by introducing a vertex kernel for getting knowledge about the unrated semantic categories starting from those categories which are known. Another interesting direction about the usage of LD

for content-based RSs is explored in [105] where the authors present Contextual eVSM, a content-based context-aware recommendation framework that adopts a semantic representation based on distributional models and entity linking techniques. In particular entity linking is used to detect entities in free text and map them to LD.
Finally, in [45] the authors propose the usage of recommendation techniques for providing personalized access to LD. The proposed method is a user-user collaborative filtering recommender wherein the similarity between the users takes into account the commonalities and informativeness of the resources instead of treating resources as plain identifiers.

8.3 Feature-based Semantic Similarity Measurement

Information resources in the Web of Data are semantically represented using RDF graphs. To evaluate the similarity between two resources, characteristics like nodes, links, and the mutual relationships are incorporated into calculation. Among others, feature-based semantic similarity metrics quantify similarity between resources in an RDF graph as a measure of commonality and distinction of their hallmarks. The features of an object can be represented in one of the following forms: binary values, nominal values, ordinal values, and cardinal values. Measuring similarity using features is based on the premise that the more common features two objects hold, the more similar they are. Bearing on this principle, feature-based semantic similarity metrics first attempt to characterize resources in an RDF graph as feature sets and then perform similarity calculation on them. In the following subsections we briefly recall the feature-based metrics for computing similarity being exploited in our evaluation. The four metrics have been chosen as representative of the feature-based similarity class since they consider different aspects of the underlying semantic graph for characterizing resources and computing similarity.

GbkSim. The authors in [111] propose a solution to compute similarity by means of a graph-based kernel. By *GbkSim*[4] an abstract walker is sent to explore the `RDF` graph to a specific depth d, en route it collects nodes and edges. The features of a resource α are represented as a vector: $\overrightarrow{a} = (w_{r_1}, w_{r_2}, .., w_{r_n})$. Each element of the vector corresponds to the weight of a resource in the feature set. The weight for resource r_i is calculated as $w_{r_i} = \sum_{m=1}^{d} \gamma_m . c_{\hat{P}^m(\alpha),r_i}$; in which the coefficient γ_m is experimentally selected upon calculation; $c_{\hat{P}^m(\alpha),r_i}$ is the number of edges that connect α to node r_i and it is calculated as: $c_{\hat{P}^m(\alpha),r_i} = |\{(r_i, r_j)|(r_i, r_j) \in \hat{P}^m(\alpha)\}|$; $\hat{P}^m(\alpha)$ is the set of edges collected at depth m. The similarity between two resources α and β is computed as the product of their corresponding feature vectors $\overrightarrow{a} = \{a_i\}_{i=1,..,n}$ and $\overrightarrow{b} = \{b_i\}_{i=1,..,n}$:

$$GbkSim(\alpha, \beta) = \frac{\sum_{i=1}^{n} a_i \times b_i}{\sqrt{\sum_{i=1}^{n}(a_i)^2} \times \sqrt{\sum_{i=1}^{n}(b_i)^2}} \tag{8.1}$$

VsmSim. In [42] an approach to characterize entities and compute similarity is introduced and evaluated. By *VsmSim*, two entities are supposed to be similar if: (i) There exist direct links between them; (ii) They point to the same object with the same property; (iii) They are pointed by the same subject with the same property. The features of a movie α corresponding to property p are the nodes connected to α through p and represented using the Vector Space Model: $\overrightarrow{a_p} = (w_{r_1,p}, w_{r_2,p}, .., w_{r_n,p})$; in which $w_{r_i,p}$ is the weight of movie r_i wrt. property p, it is computed as the `tf-idf` value of the movie: $w_{r_i,p} = f_{r_i,p} * log(\frac{M}{a_{r_i,p}})$; where $f_{r_i,p}$ is the number of occurrence of movie r_i; M is the number of movies in the collection; $a_{r_i,p}$ is the number of movies pointing to a_{r_i} via p. The similarity related to p is obtained by calculating the cosine similarity of the vectors $\overrightarrow{a_p} = \{a_{i,p}\}_{i=1,..,n}$ and $\overrightarrow{b_p} = \{b_{i,p}\}_{i=1,..,n}$:

$$VsmSim_p(\alpha, \beta) = \frac{\sum_{i=1}^{n} a_{i,p} \times b_{i,p}}{\sqrt{\sum_{i=1}^{n}(a_{i,p})^2} \times \sqrt{\sum_{i=1}^{n}(b_{i,p})^2}}$$

[4]For a clear presentation, in the scope of this paper we assign a name for the metrics that have not been named originally.

Given a set P of properties, the final similarity value can be computed as the (weighted) mean of the values computed for each property p

$$VsmSim(\alpha, \beta) = \frac{\sum_{p \in P} \omega_p VsmSim_p(\alpha, \beta)}{|P|} \tag{8.2}$$

with ω_p being weights computed via a genetic algorithm.

FuzzySim. In an attempt to incorporate the human judgment of similarity, a similarity metric, *FuzzySim* is presented in [173]. Properties are considered as features and intuitively classified into groups in descending order according to their level of importance $(g_1, g_2, .., g_n)$. The similarity value between two resources α and β on group g_i is defined as: $S_i(\alpha, \beta) = \frac{f_i(\alpha,\beta)}{f_i(\alpha)}$; where $f_i(\alpha, \beta)$ is the set of features pertaining to property group g_i that α and β have in common; $f_i(\alpha)$ is the set of features of α wrt. g_i. The membership degree of the similarity value corresponding to g_i is: $\mu(S_i) = (S_i)^{i-r(g_i,c)}$; where $r(g_i, c)$ is the ratio of the number of properties for set g_i wrt. the total number of properties. The weight $\varphi_j(m)$ for the j^{th} element of the property set is given by: $\varphi_j(m) = \sqrt{\frac{\sum_{k=1}^{j} m_k}{\sum_{k=1}^{n} m_k}} - \sqrt{\frac{\sum_{k=1}^{j-1} m_k}{\sum_{k=1}^{n} m_k}}$ in which $m = (\mu(b_1), \mu(b_1), .., \mu(b_n))$ is the ascending sorted membership vector of $(S_1, S_2, .., S_n)$. The similarity between α and β is computed by means of a fuzzy function:

$$FuzzySim(\alpha, \beta) = aggr(S_1, S_2, ..., S_n) = \sum_{j=1}^{n} b_j.\varphi_j(m) \tag{8.3}$$

Jaccard. For comparison, we use the Jaccard's index to compute similarity between feature sets. The features of a resource are modeled as a set of nodes in its surroundings. For two resources α and β, two abstract walkers are deployed to traverse the graph at a specific depth to acquire features. At each depth, a walker collects nodes, after visiting depth d, the walkers return the set of nodes $N_d(\alpha)$ and $N_d(\beta)$. The metric calculates the similarity between two resources using the Jaccard's index:

$$Jaccard(\alpha, \beta) = \frac{|N_d(\alpha) \bigcap N_d(\beta)|}{|N_d(\alpha) \bigcup N_d(\beta)|} \tag{8.4}$$

8.4 Experimental Setting

Data extracted from LD might be suitable for certain purposes but not for every purpose [79]. The quality of a piece of data is heavily dependent on the usage as well as the tasks performed on it [101]. For measuring the *fitness for use* of a dataset, a set of *quality dimensions* needs to be identified [101]. Scoring functions can be used to calculate an assessment score from the related *quality indicators* as a gauge of how well suitable the data for a particular purpose is. In the scope of this paper, we work with a specific use case, LD for the music domain used as input for recommendation tasks. Recommender systems are built to suggest things that are of interest to a user, e.g. books, movies, songs [42]. To be able to provide users with meaningful recommendations, recommender systems may enrich their background data by exploiting external sources. In this sense, the quality of the input data plays a key role in producing adequate recommendations. As seen in Section 8.2, most of the approaches to recommendation built on top of LD datasets exploits DBpedia. To our knowledge, an analysis on the influence of the underlying dataset for the quality of recommendation results has not been performed yet. Having this observation in mind, we compared recommendation results by using two of the richest encyclopedic LD sources. Data retrieved from both DBpedia and Freebase[5] is then used for computing similarity between resources employing the aforementioned similarity metrics. Afterwards, the computed similarity values are fed into a content-based recommender system to produce the final recommendations. For judging data quality, we take into account the quality dimensions of *Accuracy*, *Aggregate Diversity*, and *Novelty* in a top-N recommendation task. Recently, accuracy has been recognized to be not sufficient to evaluate a recommender system. *Aggregate Diversity* represents an important quality dimension for both business and user perspective, since improving the coverage of the items catalog and of the distributions of the items across the users may increase the sales

[5]We used the RDF version Freebase released as baseKB available at http://basekb.com/.

and the user satisfaction [160]. *Novelty* measures the ability of the system to foster discovery in the recommendation workflow [32].

(i) Considering only the top N results, for measuring *Accuracy* we use precision P@N (the fraction of the top-N recommended items being relevant to the user u) and recall R@N (the fraction of relevant items from the test set appearing in the N predicted items).

(ii) To measure *Aggregate Diversity*, we consider catalog coverage [58] (the percentage of items in the catalog that have ever been recommended to users), and Entropy and Gini coefficient [6, 160] (for the distribution of recommended items). The latter are useful to analyze the concentration degree of items across the recommendations. The scale for Gini coefficient is reversed, thereby forcing small values to represent low distributional equity and large values to represent higher equity.

(iii) One metric is chosen to measure the *Novelty* of the recommendations: the percentage of long-tail items among the recommendations across all users [6], considering the 80 percent of less rated items in the training set as *Long-tail* items.

For our experiments, we re-used the setup adopted in chapter 7. Specifically, we have implemented a content-based recommender system using a k-nearest neighbors algorithm (see Section 2.2 and Formula 2.3). Given an item-based similarity measure, INN approach estimates the rating of user u for item i using the following formula:

$$r^*(u,i) = \frac{\sum_{j \in N(i) \cap r(u)} sim(i,j) \cdot r(u,j)}{\sum_{j \in N(i) \cap r(u)} sim(i,j)}$$

where $r(u)$ represents the items rated by the user u, and $r(u,j)$ the rating value given by the user u with respect to the item i. The function $sim(\alpha, \beta)$ was computed using the similarity metrics shown in the previous section and k was fixed at 20. We selected the well-known dataset `Last.fm`

hetrec-2011. In order to compare the two LD datasets in an ordinary situation, we downsized the number of artists and bands to the 1000 most popular ones and, after that reduction, we removed the cold users, i.e. those having the number of ratings below the average of all users. The reason behind this choice was to reduce as much as possible the well known negative effect on the computation of the recommendation list due to users with a low number of ratings. After that, we used the holdout method to split the dataset into training set and test set. We built the training set by using, for each user, the first 80% of the her ratings and the remaining 20% to build the test set. Therefore, the first 80% of the ratings of each user represents her profile. One of our mapping datasets[6] was utilized to associate each item with its counterpart in DBpedia [110]. By using owl:sameAs links we were then able to retrieve Freebase mappings from the DBpedia ones.

8.5 Results Discussion

Feature sets are a prerequisite in similarity calculation for feature-based similarity metrics. It is, therefore, necessary to build a set of features for each resource. In an LD setting, building the the set of features goes through the selection of a set of RDF properties considered as relevant for the domain. For DBpedia, the top 20% most popular properties of the DBpedia ontology used in the musical domain apart from dbo:wikiPageWikiLink have been chosen, plus owl:sameAs, rdf:type and the dcterms:subject property that connects resources to categories. Table 8.1 shows the selected list of properties. Similarly, for Freebase we selected the set of 20% most popular properties connecting to resources whose type is either basekb:music.musical_group[7] or basekb:music.artist[8]. This results in 288 outgoing and 220 incoming properties. The set of properties is not listed here due to space limitations. An RDF graph consists of a huge number of edges and nodes, spreading out on

[6] http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/
[7] http://rdf.basekb.com/ns/music.musical_group
[8] http://rdf.basekb.com/ns/music.artist

Outbound	
rdf:type	dbo:associatedAct
owl:sameAs	dbo:influenced
dbo:instrument	dbo:influencedBy
dbo:writer	dbo:bandMember
dcterms:subject	dbo:formerBandMember
dbo:associatedBand	dbo:currentMember
dbo:associatedMusicalArtist	dbo:pastMember
dbo:background	dbo:occupation
dbo:genre	dbo:birthPlace

Inbound	
dbo:previousWork	dbo:producer
dbo:subsequentWork	dbo:artist
dbo:knownFor	dbo:writer
dbo:award	dbo:associatedBand
dbo:album	dbo:associatedMusicalArtist
dbo:notableWork	dbo:musicalArtist
dbo:lastAppearance	dbo:musicalBand
dbo:basedOn	dbo:musicComposer
dbo:starring	dbo:bandMember
dbo:series	dbo:formerBandMember
dbo:openingFilm	dbo:starring
dbo:related	dbo:composer

Table 8.1: The set of properties used for collecting feature sets from DBpedia.

numerous layers of predicates. It is certainly impractical to address all nodes and edges in it. Therefore, we collected a set of features by expanding the graph using the selected set of properties up to a limited depth. Considering a pair of resources that are involved in the similarity calculation, a neighborhood graph was built by expanding from each resource using the selected set of properties. For each resource, depending on the type of experiments, features can be collected in one or two levels of edges. Furthermore, also depending on the purpose of measurement, an extension can either be done using only outbound edges or using both inbound and outbound edges.
In order to investigate the effect of the selection of feature sets on the outcome, we carried out experiments using independent settings. First, we considered different levels of depth and then in each setting, the selection of properties for collecting a set of features. Two independent similarity cal-

culations have been performed: similarity computed with one-hop features and similarity computed with two-hop features. The experimental results are clarified in the following sub-sections.

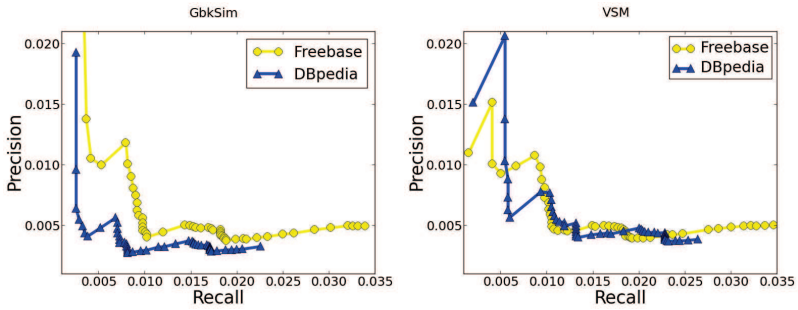
		Precision	Recall	Coverage	Entropy	Gini	%Long-tail
GbkSim	Top-10	Freebase	Freebase	Freebase	DBpedia	DBpedia	DBpedia
	Top-20	Freebase	Freebase	Freebase	DBpedia	DBpedia	DBpedia
	Top-30	Freebase	Freebase	Freebase	DBpedia	DBpedia	DBpedia
VsmSim	Top-10	Freebase	Freebase	Freebase	DBpedia	DBpedia	DBpedia
	Top-20	Freebase	DBpedia	DBpedia	DBpedia	DBpedia	DBpedia
	Top-30	Freebase	DBpedia	DBpedia	DBpedia	DBpedia	DBpedia
FuzzySim	Top-10	Freebase	Freebase	Freebase	DBpedia	DBpedia	DBpedia
	Top-20	Freebase	Freebase	Freebase	DBpedia	DBpedia	DBpedia
	Top-30	Freebase	Freebase	Freebase	DBpedia	DBpedia	DBpedia
Jaccard	Top-10	Freebase	Freebase	Freebase	Freebase	Freebase	DBpedia
	Top-20	Freebase	Freebase	Freebase	Freebase	DBpedia	DBpedia
	Top-30	Freebase	Freebase	Freebase	Freebase	Freebase	DBpedia

Table 8.2: Comparison of results for the four algorithms with Top-10, Top-20, Top-30 between `DBpedia` and `Freebase` using both inbound and outbound properties. The name in a cell indicates the dataset that obtains the best result. With largest Top-N the differences between `DBpedia` and `Freebase` are similar to the Top-30 results, therefore they are omitted due to space limitations.

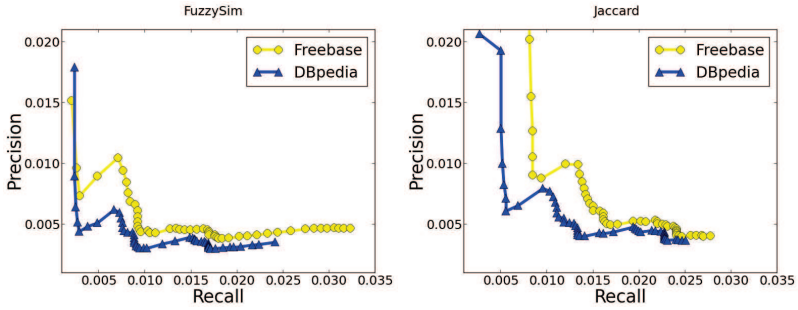
One-hop Features. Experiments were conducted in accordance with two separate configurations:

Configuration 1. *Both inbound and outbound properties are used to build the set of features of a resource.*

Accuracy. Figure 8.1 shows the precision and recall values for all metrics. Generally, recommendations computed using data extracted from `Freebase` have a better precision-recall balance and higher recall values. This holds for all similarity metrics except for *VsmSim*. Using the latter, generally there is an overlap among the values, but still `Freebase` helps achieve the highest recall values. Table 8.2 displays the quality indicators for all the metrics on both datasets considering Top-10, Top-20 and Top-30. Those results demonstrate that `Freebase` dataset brings the highest accuracy for all the similarity metrics, except for *VsmSim* as mentioned before. However, the differences between the two datasets often have a marginal significance, whereas the charts in Figure 8.1 show a more complete and general view in



(a)



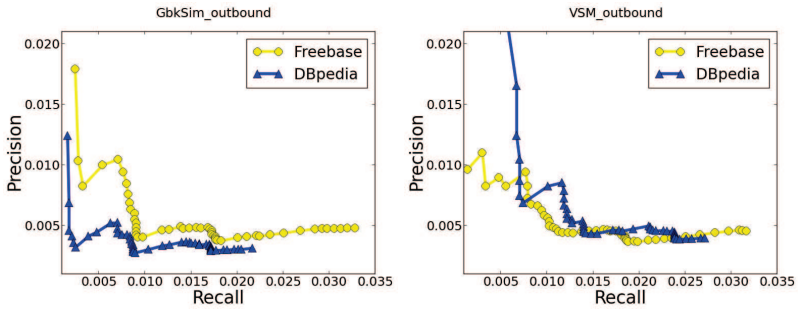
(b)

Figure 8.1: **Recommedation using similarity values computed on one-hop features**: Precision - Recall curves obtained by varying the length of the recommendations list from 1 to 50, with 20 neighbors. Inbound and outbound links are used in combination.

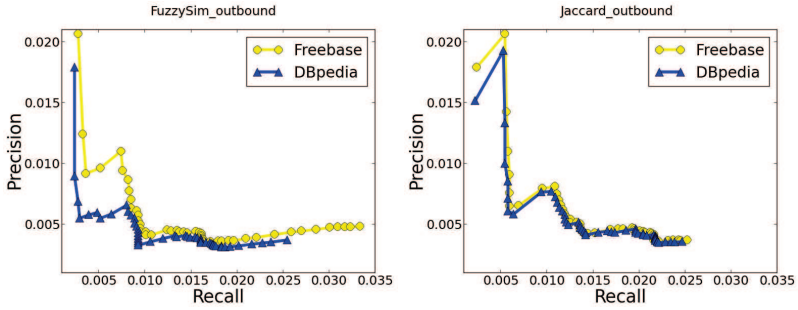
term of accuracy.

Aggregate Diversity. As shown in Table 8.2, using `Freebase` data always produces better coverage. In terms of distribution (Entropy and Gini), generally using data from `DBpedia` obtains better values compared to `Freebase`. However, those results are not easily comparable because the `DBpedia` coverage values are too low. By recommending very few items, it is much more likely to obtain a good distribution; whereas, by recommending more items, many of these may be suggested few times (even just once). This is confirmed by the fact that the entropy values are closer than the Gini values between `DBpedia` and `Freebase`, considering that Gini index is more sensible to the inequality and Entropy to the distribution among the recommendations.

Novelty. In terms of percentage of long-tail items, `DBpedia` contributes



(a)



(b)

Figure 8.2: **Recommedation using similarity values computed on one-hop features**: Precision - Recall curves obtained by varying the length of the recommendations list from 1 to 50, with 20 neighbors. Only outbound links are used.

to a better novelty compared to `Freebase` in almost every configuration. This means that using `DBpedia` tends to suggest a smaller subset of items, but these do not necessarily belong to the most popular ones. In contrast, `Freebase` can help cover more items but generally with a slightly larger popularity bias.

Configuration 2.*Only outbound properties are used to build the set of features of a resource.*

Figure 8.2 shows the accuracy obtained by the recommendations computed using similarity results in this setting. A noteworthy observation is that, for all similarity metrics, the accuracy of the recommendations calculated by using data from `DBpedia` is analogous to the accuracy obtained by using data from `Freebase`. We also observed the same trend for all metrics

by other quality dimensions (Aggregate Diversity and Novelty). Thus, the corresponding quality indicators are not depicted due to space limitations. Compared with Configuration 1, we come to the conclusion that the utilization of both inbound and outbound properties for computing semantic similarity contributes towards an improvement in the recommendation results.

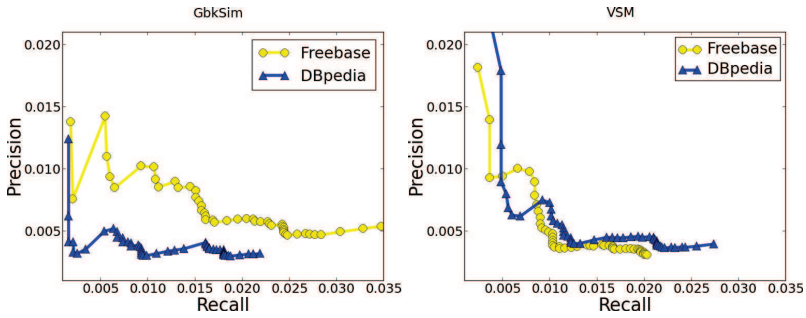
		Precision	Recall	Coverage	Entropy	Gini	%Long-tail
GbkSim	Top-10	Freebase	Freebase	Freebase	Freebase	DBpedia	DBpedia
	Top-20	Freebase	Freebase	Freebase	DBpedia	DBpedia	DBpedia
	Top-30	Freebase	Freebase	Freebase	DBpedia	DBpedia	DBpedia
VsmSim	Top-10	DBpedia	DBpedia	Freebase	Freebase	Freebase	DBpedia
	Top-20	DBpedia	DBpedia	Freebase	DBpedia	DBpedia	DBpedia
	Top-30	DBpedia	DBpedia	Freebase	Freebase	DBpedia	DBpedia
FuzzySim	Top-10	Freebase	Freebase	Freebase	Freebase	DBpedia	DBpedia
	Top-20	Freebase	Freebase	Freebase	DBpedia	DBpedia	DBpedia
	Top-30	Freebase	Freebase	Freebase	Freebase	DBpedia	DBpedia
Jaccard	Top-10	Freebase	Freebase	Freebase	DBpedia	DBpedia	Freebase
	Top-20	Freebase	Freebase	Freebase	DBpedia	DBpedia	DBpedia
	Top-30	Freebase	Freebase	Freebase	DBpedia	DBpedia	DBpedia

Table 8.3: Comparison of results for the four algorithms with Top-10, Top-20, Top-30 between DBpedia and Freebase with exploration up to two hops using both inbound and outbound properties. The name in a cell indicates the dataset that obtains the best result.

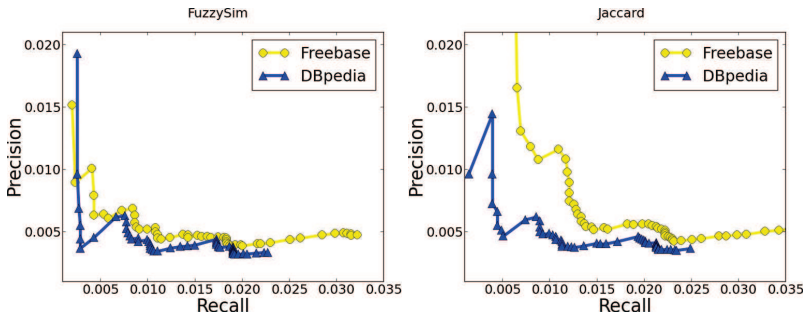
Two-hop Features. We studied the influence of exploration depth for collecting features over the recommendation outcomes. Hence, the same experimental procedures were replicated with depth $d = 2$ and the results obtained are as follows:

Configuration 1*.Both inbound and outbound properties are used*

The accuracy values for all metrics using 2 hops are depicted in Figure 8.3. Similar to the experiments performed using one-hop features, we witnessed the same pattern of the quality indicators for this experimental setting. Using the Freebase dataset to produce recommendations yields a better precision-recall balance as well as higher recall values. For both *VsmSim* and *Jaccard*, similarity values on the DBpedia dataset help produce the best recommendations in terms of accuracy; meanwhile similarity values computed by *Jaccard* on the Freebase dataset contribute to a better precision-recall balance. Considering Top-10, Top-20 and Top-30, the corresponding quality indicators for all the metrics are shown in Table 8.3. Once again, apart from *VsmSim*, rec-



(a)



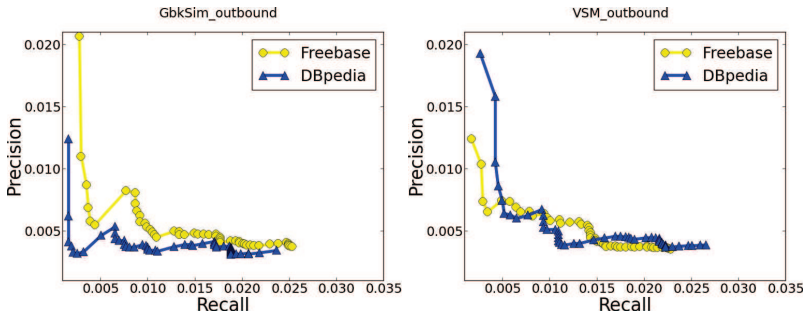
(b)

Figure 8.3: **Recommedation using similarity values computed on two-hop features**: Precision - Recall curves obtained by varying the length of the recommendations list from 1 to 50, with 20 neighbors. Inbound and outbound links are used in combination.

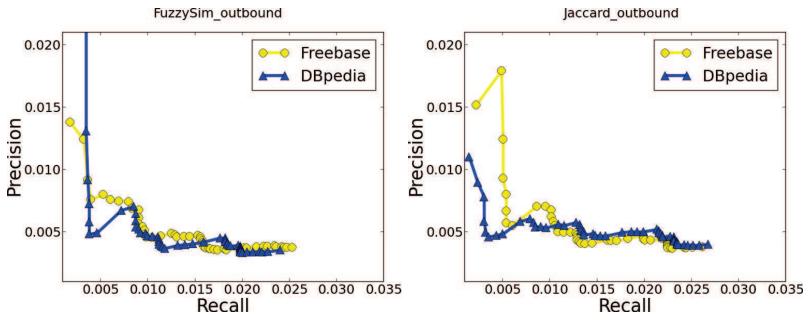
ommendation with the `Freebase` dataset using other similarity metrics still brings the highest accuracy.

Configuration 2*.Only outbound properties are used*

For this experimental setting, by all metrics we also obtained comparable results using similarity values calculated from Configuration 2 for one-hop features. Figure 8.4 depicts the precision-recall balance for all similarity metrics. The results obtained using `DBpedia` show no substantial difference compared to the results with considering also inbound properties. While the results for `Freebase` show an overall strong decrease both in terms of precision-recall balance and recall values, demonstrating that the inbound properties in `Freebase` dataset play an important role, as already seen for one-hop configuration. This decrease is particularly evident using *GbkSim*



(a)



(b)

Figure 8.4: **Recommedation using similarity values computed on two-hop features**: Precision - Recall curves obtained by varying the length of the recommendations list from 1 to 50, with 20 neighbors. Only outbound links are used.

and *Jaccard*.

It can be seen that, the outcomes of the recommendations on two-hop features confirm the experimental results for recommendation using one-hop features.

Comparison between using One-hop and Two-hop Features. We carried out a comparative analysis between using one-hop and two-hop features. As a matter of fact, the exploration of the graph comes at a price and sometime it might not be necessary. Using `DBpedia` with inbound and outbound properties, there are no relevant differences expanding the features up to two hops. Considering Figures 8.1 and 8.3, with respect to `Freebase` with inbound and outbound properties, *GbkSim* metric with two-hop features obtains better results in terms of precision with respect to one-hop configu-

ration. In terms of recall, using the *Jaccard* metric with two-hop features obtains better results with respect to one-hop configuration. Conversely, the recall values using *VsmSim* decrease with two-hop instead one-hop features. There are no substantial differences in the case of *FuzzySim*. Table 8.4 shows the gains and losses obtained expanding the features up to two hops with Top-10, Top-20 and Top-30, confirming what has been said so far. Considering the Aggregate Diversity measure, using `DBpedia` we obtain better results with two-hop features using all the similarity metrics. Using `Freebase` gains better results with two-hop features using *Jaccard* and *VsmSim*. However, `Freebase` always overcomes `DBpedia`. It is worth noticing that the recommendation distribution (Entropy and Gini measures) achieves substantial improvements with two-hop features for each configuration. Instead, when only outbound properties are used, the performances by utilizing `DBpedia` are slightly lower expanding the features up to two hops, especially in terms of precision with *VsmSim* and *FuzzySim*. With respect to `Freebase`, the recall decreases especially with *GbkSim* and *FuzzySim*. The adoption of `Freebase` instead of `DBpedia` shows its benefits when used in conjunction with *GbkSim*, when two-hop features are considered. The other similarity metrics – even though they are relatively simple – do not exhibit that considerable improvements to justify the increased computational effort needed to further explore the semantic graph of one more hop.

Discussion on General Trends

In this section we discuss the general trends emerging from Table 8.2, 8.3 and 8.4.

By looking at Table 8.2 and Table 8.3, an interesting question arises: *why* ***`Freebase`*** *seems to facilitate better accuracy and catalog coverage while* ***`DBpedia`*** *helps obtain superior novelty and aggregate diversity*[9]?

As for accuracy, we assume that in `Freebase`, at least for our target domain, items considered as similar by users are actually connected by rele-

[9] A further and more detailed investigation is needed for *VsmSim*.

vant properties with each other. This reflects the strong crowd-sourced nature of `Freebase` and also means that, in this case, `Freebase` is richer than `DBpedia` in terms of encoded knowledge. Both data sources are derived from `Wikipedia`, however `Freebase` can be flexibly edited by user communities who utilize numerous sources for encoding metadata. Thus, each `Freebase` topic consists of an expansion of the original `Wikipedia` topic, which is not the case in `DBpedia`, and specially for domains being managed by Google, `Freebase` has a higher topic coverage than `DBpedia` [88]. Moreover, the social nature of `Freebase` also implies that items resulting popular among the users are also "popular" in the underlying graph. This means that they are richer in terms of related data and are more connected to other entities. This also explains both the higher value of precision and recall and the lower values of novelty when using `Freebase`. Indeed, on the one side we know that computing recommendations based on items popularity results in good predictions for the end users [37]; on the other side, as with `Freebase` we concentrate more on popular items we have lower results when evaluating novelty (long-tail) compared to `DBpedia`. Regarding the differences between Coverage and aggregate diversity (*Entropy* and *Gini* index) a possible explanation is due to the very low values of catalog coverage when using `DBpedia`. Since there are less recommended items from the catalog, they have a higher probability to be better distributed across the users.

The results summarized in Table 8.4 show other interesting trends when exploring the underlying graph to compute recommendation. We see that values for novelty tend to decrease when we move from a one-hop to a two-hop exploration while this is not the case for catalog coverage and aggregate diversity. Possible explanations for these behaviors are: (i) popular items get more connected when exploring the graph thus obtaining better similarity results. This justifies the novelty decrease; (ii) the increasing in the number of connections also reflects in the selection of more items (better coverage) even if the new items are selected mostly among the popular ones; (iii) finally, as we have better similarity values due to better overlaps among items descrip-

			Precision	Recall	Coverage	Entropy	Gini	%Long-tail
GbkSim	Top-10	Freebase	+0.4286%	+0.4022%	-0.1557%	+0.1585%	+0.6806%	-0.015%
		DBpedia	-0.0625%	-0.0429%	+0.2697%	-0.0046%	-0.0206%	-0.0123%
	Top-20	Freebase	+0.1569%	+0.1259%	-0.0605%	+0.1214%	+0.5351%	+0.0263%
		DBpedia	+0.1379%	+0.1481%	+0.2093%	+0.0056%	+0.0258%	-0.0038%
	Top-30	Freebase	+0.2727%	+0.2582%	-0.0745%	+0.1116%	+0.5355%	-0.0012%
		DBpedia	+0.0833%	+0.0654%	+0.1864%	+0.0061%	+0.0377%	-0.015%
VsmSim	Top-10	Freebase	-0.1159%	-0.1176%	+0.1962%	+0.1094%	+0.3565%	-0.3645%
		DBpedia	-0.0423%	-0.0286%	+0.4561%	+0.108%	+0.1703%	-0.0289%
	Top-20	Freebase	-0.1957%	-0.1985%	+0.2314%	+0.0863%	+0.2964%	-0.2596%
		DBpedia	-0.0851%	-0.0758%	+0.6973%	+0.0573%	+0.0073%	-0.0885%
	Top-30	Freebase	-0.1364%	-0.1514%	+0.1709%	+0.1098%	+0.4559%	-0.2204%
		DBpedia	-0.0417%	-0.03%	+0.8538%	+0.0205%	-0.106%	-0.0379%
FuzzySim	Top-10	Freebase	-0.0492%	-0.0652%	-0.048%	+0.1195%	+0.5121%	-0.0944%
		DBpedia	+0.0392%	+0.0263%	+0.1783%	-0.0249%	-0.0004%	-0.0002%
	Top-20	Freebase	+0.0213%	+0.0224%	-0.0042%	+0.1154%	+0.5574%	-0.088%
		DBpedia	+0.1563%	+0.1319%	+0.0857%	+0.0099%	+0.0988%	+0.0297%
	Top-30	Freebase	+0.0732%	+0.0398%	-0.0218%	+0.1141%	+0.6022%	-0.0316%
		DBpedia	+0.1081%	+0.1234%	+0.02%	+0.023%	+0.1673%	-0.003%
Jaccard	Top-10	Freebase	-0.1099%	-0.1176%	+0.6162%	+0.0354%	-0.0005%	+0.0319%
		DBpedia	-0.1918%	-0.1682%	+0.2653%	+0.0896%	+0.1961%	-0.0644%
	Top-20	Freebase	-0.0714%	-0.0932%	+0.65%	-0.009%	-0.1124%	-0.1016%
		DBpedia	-0.1489%	-0.1504%	+0.3159%	+0.0859%	+0.2036%	-0.0529%
	Top-30	Freebase	0.0%	-0.0091%	+0.6874%	-0.0178%	-0.146%	-0.0558%
		DBpedia	-0.0213%	-0.0153%	+0.2858%	+0.076%	+0.1894%	-0.0082%

Table 8.4: Gains and losses obtained using two-hop features respect to one-hop ones using both inbound and outbound properties.

tions, we gain in aggregate diversity as a better similarity values means a better chance to be recommended.

8.6 Summary

In this section we have presented an investigation on the usage of DBpedia and Freebase for feeding content-based recommender systems in the music domain. In particular, we compare the recommendations obtained using separately DBpedia and Freebase using four different semantic similarity metrics. The comprehensive discussion of the experimental results provided in the previous section answers to the research questions stated in the introduction and shows that DBpedia and Freebase lead to very different results. As an answer to the question RQ1, `Freebase` seems to provide better accuracy and catalog coverage, but less novelty respect to `DBpedia`. Considering the question RQ2, `Freebase` obtains the best results with *GbkSim*, while recommendations calculated by using `DBpedia` show slightly difference among

the different similarity metrics. Finally, the comparison between the use of one-hop and two-hop features in the similarity computations responds to the question RQ3. We think that the most important outcome of such comparison regards the improvements obtained by using `Freebase` in conjuction with *GbkSim* in term of accuracy. However, in the case of using `Freebase` with the other similarity metrics or `DBpedia` regardless the similarity metric do no lead to relevant improvements that justify the computational cost of using two-hop features.

Chapter 9

Conclusions

9.1 Introduction

The final result of this thesis is a set of proposals, analysis, and discoveries in the field of Recommender Systems using Linked Data with a focus on different quality factors of recommendations, besides accuracy. First we have proposed a an adaptive multi-attribute diversification approach to personalize the recommendation diversity considering the individual inclination of the user to diversifying over different content-based item dimensions. Subsequently, we have proposed an intent-aware multi-attribute diversification approach based on regression trees. Then, we explored the analysis of temporal dynamics for a better user intent modeling, proposing a time-based analysis relying on a temporal decay function and another method based on a new technique for session analysis and tested them as intent model for the intent-aware diversification task. To tackle the user cold-start problem, we have explored the use of a number of recommendation methods following a rigorous methodology for cold-start both in single and cross-domain sce-

narios. Moreover, the impact of the size and diversity of the user profile in the source domain on the quality of the target recommendations has been evaluated in this work. Finally, we have investigated the suitability and performance of two graph-based similarity metrics - SimRank and PageRank - to feed a content-based recommender system. Moreover, a comparative analysis of the two well-known Linked Data resources DBpedia and Freebase for feeding a content-based recommender system has been carried out.

9.2 Summary and Contributions

This section summarizes the main findings and contributions of this thesis.

9.2.1 Adaptive Multi-Attribute Diversity

In Chapter 3, pursuing the research goals RG1 and RG2, we have proposed a novel adaptive multi-attribute diversification method following the intuition that users with more diverse items in the profile might prefer diverse recommendations. The method models the user profile by taking into account her attitude to enjoy (or not) items which result diverse with regard to different attributes and eventually adopt this model to diversify recommendation list in a personalized fashion. The modeling strategy relies on a user classification for the different descriptive attributes of the items. We have suggested two different versions of our profile modeling (we called them hard and fuzzy). Our model can be used in both implicit and explicit diversification methods, and hence we built them upon two different state-of-the-art diversification methods, namely `MMR` and `xQuAD`, that are respectively

Our experiments validate the proposed approach, showing its ability to overcome the traditional accuracy-diversity trade-off issue, improving different quality objectives, without affecting the others.

9.2.2 Regression Trees for Multi-Attribute Diversity

In Chapter 4, pursuing the research goal RG1, we have proposed an intent-aware diversification method that leverages regression trees as user modeling technique in a multi-attribute scenario. Compared to approaches where item attributes are treated independently one to each other, regression trees make possible to represent user tastes as a combination of interrelated characteristics, since in a regression tree conditional probability lets to build inference rules about user's preferences regards different item's attributes. We have moreover analyzed the combine in sequence of different diversification algorithms by means of a two phase re-ranking procedure, with the aim of benefiting from the strengths of both.

The analysis of the experimental results suggests that a pure rule-based diversification is a good choice when the aggregate diversity is more needed than individual diversity. Conversely, basic `xQuAD` remains the best choice to improve the individual diversity while its combination with the rule-based diversification improves also the aggregate diversity.

9.2.3 Diversification with Temporal Dynamics

In Chapter 5, pursuing the research goal RG3, we have proposed two different time-dependent user modelings which take into account also the user rating history. The intuition behind our idea is that temporal dynamics might allow to better understand the user interests with respect to the items characteristics and then provide a more accurate intent-aware diversification. We have presented two intent modeling methods based on temporal dynamics. The first one analyses the frequency of interaction between the users and the items features using a temporal decay function, while the other method relies on a novel session analysis technique for intent modeling.

The experimental results show that the analysis of temporal dynamics leads to better accuracy-diversity balance and intent-aware diversity, but only by means of our new session analysis technique. Moreover, our method leads

to better aggregate diversity of recommendations, demonstrating a higher degree of personalization among the users.

9.2.4 LOD and Cross-Domain For Cold-Start

In Chapter 6, pursuing the research goal RG4, we have carried out a comparison of different hybrid recommendation methods that jointly exploit user ratings and item metadata extracted from Linked Data for tackling the used cold start problem, in both single and cross-domain scenarios. In particular, we have evaluated a number of memory-based, matrix factorization based, and graph-based algorithms in terms of accuracy, individual diversity and catalog coverage. Finally, we have analysed the impact of user profile characteristics, i.e. size and diversity, on the cross-domain recommendation accuracy.

The experimental results show the benefits of cross-domain information in cold-start situations in terms of ranking accuracy. Moreover, the results demonstrated that by exploiting item metadata, when cross-domain information is not available, the graph-based methods are able to provide relevant recommendations even for users with very few likes. Regarding diversity we observe different behaviour in the two datasets, and therefore conclude that in general the results depend on the target domain. We have also studied the impact of the size and diversity of user profiles in the source domain, concluding that while more cross-domain user preferences are helpful, a greater item diversity in the source profile can actually harm the performance in the target domain.

9.2.5 Graph-based Similarity Metrics

In Chapter 7, pursuing the research goal RG5, we have reviewed the graph-based similarity metrics SimRank and PageRank, and investigated their performance for computing similarity between resources in Linked Data sources for the content-based recommendation task, against two state-of-the-art content-

based algorithms able to exploit metadata extracted from LD sources.

The experimental results show that SimRank and Personalized PageRank can produce interesting results compared to the two baselines in terms of accuracy and novelty, while they tend to penalize the aggregate diversity of the recommendations.

9.2.6 Comparative Analysis of DBpedia and Freebase

In Chapter 8, pursuing the research goal RG5, we have investigated the use of the knowledge available in the two Linked Data sources DBpedia and Freebase. The choice of one of the two datasets may influence the performance of a recommendation engine not only in terms of precision of the results but also in terms of their diversity and novelty. In particular, we tested four different feature-based similarity metrics exploiting both DBpedia and Freebase in the music domain.

The analysis of the experimental results shows relevant differences between the two sources. In particular, the use of Freebase leads to better accuracy and catalog coverage while DBpedia helps obtain superior novelty and aggregate diversity.

9.3 Future Work

The work presented in this thesis opens up various possible lines of works related to the area of recommends systems.

Regarding Chapter 3, the proposed approach can be adapted to determinate the user propensity toward novelty instead of diversity, since it results another important quality dimension for the user experience. From the point of view of attributes selection, other domain independent side information may be taken into account such as popularity or even latent dimensions. A further related aspect to be considered is that of time-aware selection of attributes and corresponding values. Interesting results to estimate and detect peaks of interest have already been presented in [77] while in [74] the

idea to model individual needs is put forward with respect to the novelty property, with emphasis on user's dynamic behaviour and time dependency. Additional investigation to understand the approach that should be used for users with a small profile and a high value of entropy needs also to be done together with the role of individual diversity in cold-start situations. Reasonably, a hybrid system like the one used in [179], able to switch between different approaches depending on the actual needs, could be conveniently applied to match the demand of both cold and expert users.

Another important line of future work consists in the definition of a unified adaptive diversification framework able to take into account the context information in addition to the user predilection towards diversity (Chapter 3), the multi-dimensional intents (Chapter 4), and the temporal dynamics (Chapter 5).

Regarding the work done on the user cold-start problem (Chapter 6), further experimentations can be done considering other domains and including other recommendations methods. Moreover, an exhaustive analysis of the datasets characteristics could explain the differences we found between the music and movies domains, and also among the various recommendation methods. Another interesting line of future work consists in the analysis of the usefulness of the diversification of the recommendations for cold-star users. While it has been demonstrated that diversity has a positive impact on the user satisfaction for users with at least 15 ratings [46], we are not aware of any prior work assessing such correlation for users with much less or even no feedback.

Finally, another future work regards the online evaluation of the methods proposed and explored in this thesis. As offline evaluations do not provide information about the real users' satisfaction, user studies can be useful to complement the results obtained in this thesis and show further insights.

Bibliography

[1] P. Adamopoulos and A. Tuzhilin. On over-specialization and concentration bias of recommendations: Probabilistic neighborhood selection in collaborative filtering systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 153–160, New York, NY, USA, 2014. ACM.

[2] P. Adamopoulos and A. Tuzhilin. On unexpectedness in recommender systems: Or how to better expect the unexpected. *ACM Transactions on Intelligent Systems and Technology*, 5(4):54:1–54:32, Dec. 2014.

[3] P. Adamopoulos and E. Tuzhilin. On unexpectedness in recommender systems: Or how to expect the unexpected. In *in Proc of RecSys11 Intl. Workshop on Novelty and Diversity in Recommender Systems*, 2011.

[4] R. P. Adams, G. E. Dahl, and I. Murray. Incorporating side information in probabilistic matrix factorization with gaussian processes. In P. Grnwald and P. Spirtes, editors, *UAI*, pages 1–9. AUAI Press, 2010.

[5] G. Adomavicius and Y. Kwon. *Maximizing aggregate recommendation diversity: A graph-theoretic approach*, volume 816, pages 3–10. CEUR-WS, 2011.

[6] G. Adomavicius and Y. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Trans. on Knowl. and Data Eng.*, 24(5):896–911, May 2012.

[7] G. Adomavicius and Y. Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):896–911, May 2012.

[8] G. Adomavicius and Y. Kwon. *Multi-Criteria Recommender Systems*, pages 847–880. Springer US, Boston, MA, 2015.

[9] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, June 2005.

[10] E. Agirre, M. Cuadros, G. Rigau, and A. Soroa. Exploring knowledge bases for similarity. In N. C. C. Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA).

[11] E. Agirre and A. Soroa. Personalizing pagerank for word sense disambiguation. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '09, pages 33–41, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[12] M. Aharon, O. Anava, N. Avigdor-Elgrabli, D. Drachsler-Cohen, S. Golan, and O. Somekh. Excuseme: Asking users to help in item cold-start recommendations. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 83–90, New York, NY, USA, 2015. ACM.

[13] A. Ashkan, B. Kveton, S. Berkovsky, and Z. Wen. Optimal greedy diversity for recommendation. In Q. Yang and M. Wooldridge, editors, *IJCAI*, pages 1742–1748. AAAI Press, 2015.

[14] V. A. A. Ayala, M. Przyjaciel-Zablocki, T. Hornung, A. Schätzle, and G. Lausen. Extending sparql for recommendations. In *Proceedings of Semantic Web Information Management on Semantic Web Information Management*, SWIM'14, pages 1:1–1:8, New York, NY, USA, 2014. ACM.

[15] A. B. Barragns-Martnez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-Lpez, F. A. Mikic-Fonte, and A. Peleteiro. A hybrid content-based and d collaborative filtering approach to recommend tv programs enhanced with singular value decomposition. *Information Sciences*, 180(22):4290 – 4311, 2010.

[16] F. Belém, R. Santos, J. Almeida, and M. Gonçalves. Topic diversity in tag recommendation. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 141–148, New York, NY, USA, 2013. ACM.

[17] A. Bellogín, I. Cantador, and P. Castells. A comparative study of heterogeneous item recommendations in social systems. *Information Sciences*, 221:142–169, Feb. 2013.

[18] A. Bellogin, P. Castells, and I. Cantador. Precision-oriented evaluation of recommender systems: An algorithmic comparison. In *ACM RecSys '11*, pages 333–336, 2011.

[19] J. Bennett, S. Lanning, and N. Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[20] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst*, 5(3):1–22, 2009.

[21] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutirrez. Recommender systems survey. *Knowledge-Based Systems*, 46:109 – 132, 2013.

[22] D. Bollen, B. P. Knijnenburg, M. C. Willemsen, and M. Graus. Understanding choice overload in recommender systems. In *Proceedings*

of the Fourth ACM Conference on Recommender Systems, RecSys '10, pages 63–70, New York, NY, USA, 2010. ACM.

[23] K. Bradley and B. Smyth. Improving Recommendation Diversity. In *Irish Conference in Artificial Intelligence and Cognitive Science*, pages 75–84, 2001.

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

[25] R. D. Burke. Hybrid recommender systems: Survey and experiments. *User Model. User-Adapt. Interact.*, 12(4):331–370, 2002.

[26] I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi. Cross-domain recommender systems. In *Recommender Systems Handbook*, pages 919–959. Springer, 2015.

[27] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, ICML '07, pages 129–136, New York, NY, USA, 2007. ACM.

[28] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, 1998.

[29] P. Castells, N. J. Hurley, and S. Vargas. Novelty and diversity in recommender systems. In *Recommender Systems Handbook*, pages 881–918. Springer US, Boston, MA, 2015.

[30] P. Castells, S. Vargas, and J. Wang. Novelty and Diversity Metrics for Recommender Systems: Choice, Discovery and Relevance. In *International Workshop on Diversity in Document Retrieval (DDR 2011) at the 33rd European Conference on Information Retrieval (ECIR 2011)*, Apr. 2011.

[31] D. Ceccarelli, C. Lucchese, S. Orlando, R. Perego, and S. Trani. Dexter: An open source framework for entity linking. In *Proceedings of the Sixth International Workshop on Exploiting Semantic Annotations in Information Retrieval*, ESAIR '13, pages 17–20, New York, NY, USA, 2013. ACM.

[32] O. Celma and P. Herrera. A new approach to evaluating novel recommendations. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 179–186, 2008.

[33] O. Chapelle, S. Ji, C. Liao, E. Velipasaoglu, L. Lai, and S.-L. Wu. Intent-based diversification of web search results: Metrics and algorithms. *Inf. Retr.*, 14(6):572–592, Dec. 2011.

[34] L. Chen, W. Wu, and L. He. How personality influences users' needs for recommendation diversity? In *Extended Abstracts on Human Factors in Computing Systems*, CHI EA '13, pages 829–834, 2013.

[35] C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of SIGIR '08*, SIGIR '08, pages 659–666. ACM, 2008.

[36] P. Covington, J. Adams, and E. Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 191–198, New York, NY, USA, 2016. ACM.

[37] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 39–46, New York, NY, USA, 2010. ACM.

[38] P. Cremonesi, R. Turrin, E. Lentini, and M. Matteucci. An evaluation methodology for collaborative recommender systems. 2010.

[39] M. de Gemmis, P. Lops, C. Musto, F. Narducci, and G. Semeraro. *Semantics-Aware Content-Based Recommender Systems*, pages 119–159. Springer US, Boston, MA, 2015.

[40] C. Desrosiers and G. Karypis. *A Comprehensive Survey of Neighborhood-based Recommendation Methods*, pages 107–144. Springer US, Boston, MA, 2011.

[41] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker. Linked open data to support content-based recommender systems. In *Proceedings of the 8th International Conference on Semantic Systems*, I-SEMANTICS '12, pages 1–8, New York, NY, USA, 2012. ACM.

[42] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker. Linked open data to support content-based recommender systems. In *Proceedings of the 8th International Conference on Semantic Systems*, I-SEMANTICS '12, pages 1–8, New York, NY, USA, 2012. ACM.

[43] T. Di Noia, V. C. Ostuni, J. Rosati, P. Tomeo, and E. Di Sciascio. An analysis of users' propensity toward diversity in recommendations. In *ACM RecSys '14*, RecSys '14, pages 285–288. ACM, 2014.

[44] T. Di Noia, V. C. Ostuni, J. Rosati, P. Tomeo, and E. Di Sciascio. An analysis of users' propensity toward diversity in recommendations. In *RecSys '14*, pages 285–288, 2014.

[45] M. Dojchinovski and T. Vitvar. Personalised access to linked data. In *EKAW*, pages 121–136, 2014.

[46] M. D. Ekstrand, F. M. Harper, M. C. Willemsen, and J. A. Konstan. User perception of differences in recommender algorithms. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 161–168, New York, NY, USA, 2014. ACM.

[47] M. Enrich, M. Braunhofer, and F. Ricci. Cold-start management with cross-domain collaborative filtering and tags. In *EC-Web '13*, pages 101–112, 2013.

[48] F. Eric Miller. RDF Primer. http://www.w3.org/TR/rdf-primer, 2004.

[49] I. Fernández-Tobías, M. Braunhofer, M. Elahi, F. Ricci, and I. Cantador. Alleviating the new user problem in collaborative filtering by exploiting personality information. *User Model. User-Adapt. Interact.*, 26(2-3):221–255, 2016.

[50] I. Fernández-Tobías, I. Cantador, M. Kaminskas, and F. Ricci. A generic semantic-based framework for cross-domain recommendation. In *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems*, HetRec '11, pages 25–32, New York, NY, USA, 2011. ACM.

[51] I. Fernández-Tobías, P. Tomeo, I. Cantador, T. Di Noia, and E. Di Sciascio. Accuracy and diversity in cross-domain recommendations for cold-start users with positive-only feedback. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 119–122, New York, NY, USA, 2016. ACM.

[52] P. Ferragina and U. Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1625–1628, New York, NY, USA, 2010. ACM.

[53] D. Fleder and K. Hosanagar. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management science*, 55(5):697–712, 2009.

[54] A. Freitas, J. a. Oliveira, S. O'Riain, E. Curry, and J. a. Pereira da Silva. Treo: Best-Effort Natural Language Queries over Linked Data.

In R. Muñoz, A. Montoyo, and E. Métais, editors, *Proceedings of the 16th International Conference on Applications of Natural Language to Information Systems, NLDB 2011 (poster)*, volume 6716 of *Lecture Notes in Computer Science*, pages 286–289, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[55] A. Freitas, J. a. G. Oliveira, S. O'Riain, E. Curry, and J. a. C. P. Da Silva. Querying linked data using semantic relatedness: A vocabulary independent approach. In *Proceedings of the 16th International Conference on Natural Language Processing and Information Systems*, NLDB'11, pages 40–51, Berlin, Heidelberg, 2011. Springer-Verlag.

[56] S. Funk. Netflix update: Try this at home. In *http://sifter.org/ simon/journal/20061211.html*, 2006.

[57] M. Ge, C. Delgado-battenfeld, and D. Jannach. Beyond accuracy: evaluating recommender systems by coverage and serendipity. In *In RecSys 10*, page 257, 2010.

[58] M. Ge, C. Delgado-Battenfeld, and D. Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 257–260, New York, NY, USA, 2010. ACM.

[59] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 69–77, New York, NY, USA, 2011. ACM.

[60] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *Proceedings of the 18th International Conference on World Wide Web*, WWW '09, pages 381–390, New York, NY, USA, 2009. ACM.

[61] A. Gunawardana and C. Meek. A unified approach to building hybrid recommender systems. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 117–124, New York, NY, USA, 2009. ACM.

[62] I. A. Hameed. Using gaussian membership functions for improving the reliability and robustness of students' evaluation systems. *Expert Syst. Appl.*, 38(6):7135–7142, June 2011.

[63] T. H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02, pages 517–526, New York, NY, USA, 2002. ACM.

[64] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool, 1st edition, 2011.

[65] B. Heitmann and C. Hayes. Using linked data to build open, collaborative recommender systems. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*, 2010.

[66] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and C. Zhu. Personalized recommendation via cross-domain triadic factorization. In *WWW '13*, pages 595–606, 2013.

[67] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Washington, DC, USA, 2008. IEEE Computer Society.

[68] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM '08*, pages 263–272, 2008.

[69] N. Hurley and M. Zhang. Novelty and diversity in top-n recommendation – analysis and evaluation. *ACM TOIT*, 10(4):14:1–14:30, 2011.

[70] R.-H. Hwang, Y.-L. Hsueh, and Y.-T. Chen. An effective taxi recommender system based on a spatio-temporal factor analysis model. *Information Sciences*, 314(C):28–40, Sept. 2015.

[71] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 538–543, New York, NY, USA, 2002. ACM.

[72] K. Ji, R. Sun, W. Shu, and X. Li. Next-song recommendation with temporal dynamics. *Know.-Based Syst.*, 88(C):134–143, Nov. 2015.

[73] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of the Int'l. Conf. on Research in Computational Linguistics*, pages 19–33, 1997.

[74] K. Kapoor, V. Kumar, L. Terveen, J. A. Konstan, and P. Schrater. "i like to explore sometimes": Adapting to dynamic user novelty preferences. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, 2015.

[75] A. Karatzoglou, L. Baltrunas, and Y. Shi. Learning to rank for recommender systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 493–494, New York, NY, USA, 2013. ACM.

[76] H. Khrouf and R. Troncy. Hybrid event recommendation using linked data and user diversity. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 185–192, New York, NY, USA, 2013. ACM.

[77] H. Khrouf and R. Troncy. Hybrid event recommendation using linked data and user diversity. In *ACM RecSys '13*, pages 185–192, 2013.

[78] D. Kluver and J. A. Konstan. Evaluating recommender behavior for new users. In *RecSys '14*, pages 121–128, 2014.

[79] M. Knuth, D. Kontokostas, and H. Sack. Linked data quality: Identifying and tackling the key challenges. In M. Knuth, D. Kontokostas, and H. Sack, editors, *Proceedings of the 1st Workshop on Linked Data Quality co-located with 10th International Conference on Semantic Systems, LDQ@SEMANTiCS 2014, Leipzig, Germany, September 2nd, 2014.*, volume 1215 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[80] N. Koenigstein, G. Dror, and Y. Koren. Yahoo! music recommendations: Modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, pages 165–172, New York, NY, USA, 2011. ACM.

[81] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 447–456, New York, NY, USA, 2009. ACM.

[82] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[83] O. Küçüktunç, E. Saule, K. Kaya, and U. V. Çatalyürek. Diversifying citation recommendations. *ACM Transactions on Intelligent Systems and Technology*, 5(4):55:1–55:21, Dec. 2014.

[84] N. Lathia, S. Hailes, L. Capra, and X. Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 210–217, 2010.

[85] S. Lee, S. Park, M. Kahng, and S.-g. Lee. Pathrank: A novel node ranking measure on a heterogeneous graph for recommender systems. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1637–1641, New York, NY, USA, 2012. ACM.

[86] S. Lee, S. Park, M. Kahng, and S.-g. Lee. Pathrank: A novel node ranking measure on a heterogeneous graph for recommender systems. In *CIKM '12*, pages 1637–1641, 2012.

[87] S. K. Lee, Y. H. Cho, and S. H. Kim. Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations. *Information Sciences*, 180(11):2142 – 2155, 2010.

[88] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 6, 2015.

[89] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[90] T.-Y. Liu. Learning to rank for information retrieval. *Found. Trends Inf. Retr.*, 3(3):225–331, Mar. 2009.

[91] T. Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *SIGIR '07: Proceedings of the Learning to Rank workshop in the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007.

[92] A. Lommatzsch, T. Plumbaum, and S. Albayrak. A linked dataverse knows better: Boosting recommendation quality using semantic knowledge. In *Proc. of the 5th Intl. Conf. on Advances in Semantic Processing*, pages 97 – 103, Wilmington, DE, USA, 2011. IARIA.

[93] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira,

and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[94] A. Maksai, F. Garcin, and B. Faltings. Predicting online performance of news recommender systems through richer evaluation metrics. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 179–186, New York, NY, USA, 2015. ACM.

[95] N. Marie, O. Corby, F. Gandon, and M. Ribière. Composite interests' exploration thanks to on-the-fly linked data spreading activation. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, HT '13, pages 31–40, New York, NY, USA, 2013. ACM.

[96] C. Martinez-Cruz, C. Porcel, J. Bernab-Moreno, and E. Herrera-Viedma. A model to represent users trust in recommender systems using ontologies and fuzzy linguistic modeling. *Information Sciences*, 311:102 – 118, 2015.

[97] D. G. Mcdonald and J. Dimmick. The conceptualization and measurement of diversity. *Communication Research*, 30(1):60–79, 2003.

[98] S. M. McNee, J. Riedl, and J. A. Konstan. Being accurate is not enough: How accuracy metrics have hurt recommender systems. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1097–1101, 2006.

[99] P. Melville, R. J. Mooney, and R. Nagarajan. Content-boosted collaborative filtering for improved recommendations. In *Eighteenth National Conference on Artificial Intelligence*, pages 187–192, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[100] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 1–8, New York, NY, USA, 2011. ACM.

[101] P. N. Mendes, H. Mühleisen, and C. Bizer. Sieve: Linked data quality assessment and fusion. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, EDBT-ICDT '12, pages 116–123, New York, NY, USA, 2012. ACM.

[102] A. Moro, A. Raganato, and R. Navigli. Entity Linking meets Word Sense Disambiguation : a Unified Approach. *Transactions of the Association for Computational Linguistics (TACL)*, 2014.

[103] C. Musto, P. Basile, M. de Gemmis, P. Lops, G. Semeraro, and S. Rutigliano. Automatic selection of linked open data features in graph-based recommender systems. In T. Bogers and M. Koolen, editors, *CBRecSys@RecSys*, volume 1448 of *CEUR Workshop Proceedings*, pages 10–13. CEUR-WS.org, 2015.

[104] C. Musto, P. Lops, P. Basile, M. de Gemmis, and G. Semeraro. Semantics-aware graph-based recommender systems exploiting linked open data. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization*, UMAP '16, pages 229–237, New York, NY, USA, 2016. ACM.

[105] C. Musto, G. Semeraro, P. Lops, and M. de Gemmis. Combining distributional semantics and entity linking for context-aware content-based recommendation. In *User Modeling, Adaptation, and Personalization - 22nd International Conference, UMAP 2014, Aalborg, Denmark, July 7-11, 2014. Proceedings*, pages 381–392, 2014.

[106] X. Ning, C. Desrosiers, and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender Systems Handbook*. Springer, 2015.

[107] X. Ning and G. Karypis. Sparse linear methods with side information for top-n recommendations. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 155–162, New York, NY, USA, 2012. ACM.

[108] T. D. Noia, V. C. Ostuni, P. Tomeo, and E. D. Sciascio. Sprank: Semantic path-based ranking for top-n recommendations using linked open data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):9:1–9:34, 2016.

[109] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *RecSys '13*, pages 85–92, 2013.

[110] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 85–92, New York, NY, USA, 2013. ACM.

[111] V. C. Ostuni, T. Di Noia, R. Mirizzi, and E. Di Sciascio. A linked data recommender system using a neighborhood-based graph kernel. In *The 15th International Conference on Electronic Commerce and Web Technologies*, Lecture Notes in Business Information Processing. Springer-Verlag, 2014.

[112] V. C. Ostuni, T. D. Noia, E. D. Sciascio, and R. Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *ACM RecSys '13*, pages 85–92, 2013.

[113] V. C. Ostuni, S. Oramas, T. Di Noia, X. Serra, and E. Di Sciascio. Sound and music recommendation with knowledge graphs. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2016. To appear.

[114] U. Panniello, A. Tuzhilin, and M. Gorgoglione. Comparing context-aware recommender systems in terms of accuracy and diversity. *User Modeling and User-Adapted Interaction*, 24(1-2):35–65, Feb. 2014.

[115] A. Passant. dbrec: music recommendations using dbpedia. In *Proc. of 9th Int. Sem. Web Conf.*, ISWC'10, pages 209–224, 2010.

[116] A. Passant. Measuring semantic distance on linking data and using it for resources recommendations. In *AAAI Spring Symposium: Linked Data Meets Artificial Intelligence*. AAAI, 2010.

[117] I. Porteous, A. Asuncion, and M. Welling. Bayesian matrix factorization with side information and dirichlet process mixtures. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, pages 563–568. AAAI Press, 2010.

[118] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query, 2008.

[119] A. Ragone, P. Tomeo, C. Magarelli, T. Di Noia, M. Palmonari, A. Maurino, and E. Di Sciascio. Schema-summarization in linked-data-based feature selection for recommender systems. In *32nd ACM SIGAPP Symposium On Applied Computing*. ACM, 2017.

[120] S. Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 995–1000, Washington, DC, USA, 2010. IEEE Computer Society.

[121] S. Rendle. Factorization machines. In *ICDM*, pages 995–1000. IEEE Computer Society, 2010.

[122] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

[123] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 635–644, New York, NY, USA, 2011. ACM.

[124] P. Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'95, pages 448–453, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[125] M. T. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani. Pareto-efficient hybridization for multi-objective recommender systems. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 19–26. ACM, 2012.

[126] M. T. Ribeiro, N. Ziviani, E. S. D. Moura, I. Hata, A. Lacerda, and A. Veloso. Multiobjective pareto-efficient approaches for recommender systems. *ACM Transactions on Intelligent Systems and Technology*, 5(4):53:1–53:20, Dec. 2014.

[127] F. Ricci, L. Rokach, and B. Shapira, editors. *Recommender Systems Handbook*. Springer, 2015.

[128] F. Ricci, L. Rokach, and B. Shapira. Recommender systems: Introduction and challenges. In *Recommender Systems Handbook*. Springer, 2015.

[129] G. Rizzo and R. Troncy. Nerd: A framework for unifying named entity recognition and disambiguation extraction tools. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 73–76, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[130] M. Rowe. Semanticsvd++: incorporating semantic taste evolution for predicting ratings. In *2014 IEEE/WIC/ACM International Conferences on Web Intelligence, WI 2014*, 2014.

[131] M. Rowe. Transferring semantic categories with vertex kernels: recommendations with semanticsvd++. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference*, 2014.

[132] N. Rubens, M. Elahi, M. Sugiyama, and D. Kaplan. Active learning in recommender systems. In *Recommender Systems Handbook*, pages 809–846. Springer, 2015.

[133] S. Sahebi and P. Brusilovsky. It takes two to tango: An exploration of domain pairs for cross-domain collaborative filtering. In *RecSys '15*, pages 131–138, 2015.

[134] A. Said, B. Fields, B. J. Jain, and S. Albayrak. User-centric evaluation of a k-furthest neighbor collaborative filtering recommender algorithm. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 1399–1408. ACM, 2013.

[135] R. Santos, C. Macdonald, and I. Ounis. Exploiting query reformulations for web search result diversification. In *WWW '10*, pages 881–890, 2010.

[136] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, 2001.

[137] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. pages 285–295, 2001.

[138] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl. Application of dimensionality reduction in recommender system - a case study. In *In ACM WebKDD Workshop*, 2000.

[139] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR '02*, pages 253–260, 2002.

[140] A. Schlicker, F. S. Domingues, J. Rahnenfhrer, and T. Lengauer. A new measure for functional similarity of gene products based on gene ontology. *BMC Bioinformatics*, 7:302, 2006.

[141] B. Schwartz. *The Paradox of Choice: Why More Is Less*. Harper Perennial, January 2005.

[142] S. Sedhain, S. Sanner, D. Braziunas, L. Xie, and J. Christensen. Social collaborative filtering for cold-start recommendations. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 345–348, New York, NY, USA, 2014. ACM.

[143] B. Shao, T. Li, and M. Ogihara. Quantify music artist similarity based on style and mood. In *Proceedings of the 10th ACM Workshop on Web Information and Data Management*, WIDM '08, pages 119–124, New York, NY, USA, 2008. ACM.

[144] W. Shen, J. Wang, P. Luo, and M. Wang. Linden: Linking named entities with knowledge base via semantic knowledge. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 449–458, New York, NY, USA, 2012. ACM.

[145] C. Shi, X. Kong, Y. Huang, P. S. Yu, and B. Wu. Hetesim: A general framework for relevance measure in heterogeneous networks. *CoRR*, abs/1309.7393, 2013.

[146] Y. Shi, X. Zhao, J. Wang, M. Larson, and A. Hanjalic. Adaptive diversification of recommendation results via latent factor portfolio. In *ACM SIGIR '12*, pages 175–184, 2012.

[147] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 650–658, New York, NY, USA, 2008. ACM.

[148] B. Smyth and P. McClave. Similarity vs. diversity. In *Proceedings of the 4th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development*, ICCBR '01, pages 347–361, London, UK, UK, 2001. Springer-Verlag.

[149] C. Speier, J. S. Valacich, and I. Vessey. The Influence of Task Interruption on Individual Decision Making: An Information Overload Perspective. *Decision Sciences*, 30(2):337–360, Mar. 1999.

[150] M. Stankovic, W. Breitfuss, and P. Laublet. Linked-data based suggestion of relevant topics. In *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, pages 49–55, New York, NY, USA, 2011. ACM.

[151] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. In *In VLDB 11*, 2011.

[152] N. Tintarev and J. Masthoff. Similarity for news recommender systems. In *In Proceedings of the AH06 Workshop on Recommender Systems and Intelligent User Interfaces*, 2006.

[153] P. Tomeo, T. Di Noia, M. de Gemmis, P. Lops, G. Semeraro, and E. Di Sciascio. Exploiting regression trees as user models for intent-aware multi-attribute diversity. In *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015.*, pages 2–9, 2015.

[154] P. Tomeo, I. Fernández-Tobías, T. Di Noia, and I. Cantador. Exploiting linked open data in cold-start recommendations with positive-only feedback. In *CERI '16*, 2016.

[155] P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *J. Artif. Int. Res.*, 37(1):141–188, Jan. 2010.

[156] S. Vargas, L. Baltrunas, A. Karatzoglou, and P. Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 209–216, 2014.

[157] S. Vargas, L. Baltrunas, A. Karatzoglou, and P. Castells. Coverage, redundancy and size-awareness in genre diversity for recommender systems. In *RecSys '14*, pages 209–216, 2014.

[158] S. Vargas and P. Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *ACM RecSys '11*, pages 109–116, 2011.

[159] S. Vargas and P. Castells. Exploiting the diversity of user preferences for recommendation. In *OAIR '13*, pages 129–136, 2013.

[160] S. Vargas and P. Castells. Improving sales diversity by recommending users to items. In *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 145–152, 2014.

[161] S. Vargas and P. Castells. Improving sales diversity by recommending users to items. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, 2014.

[162] S. Vargas, P. Castells, and D. Vallet. Intent-oriented diversity in recommender systems. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 1211–1212, New York, NY, USA, 2011. ACM.

[163] Y. Wang and I. H. Witten. Induction of model trees for predicting continuous classes. In *Poster papers of the 9th European Conference on Machine Learning*. Springer, 1997.

[164] J. Wasilewski and N. Hurley. Intent-aware diversification using a constrained plsa. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 39–42, New York, NY, USA, 2016. ACM.

[165] R. S. Wills. Google's pagerank: The math behind the search engine. *Math. Intelligencer*, pages 6–10, 2006.

[166] P. Winoto and T. Y. Tang. If you like the devil wears prada the book, will you also enjoy the devil wears prada the movie? A study of cross-domain recommendations. *New Generation Computing*, 26(3):209–225, 2008.

[167] W. Wu, L. Chen, and L. He. Using personality to adjust diversity in recommender systems. In *ACM HT '13*, pages 225–229, 2013.

[168] C. Yu, L. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: Diversification in recommender systems. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '09, pages 368–378, New York, NY, USA, 2009. ACM.

[169] C. Yu, L. Lakshmanan, and S. Amer-Yahia. It takes variety to make a world: Diversification in recommender systems. In *EDBT '09*, pages 368–378, 2009.

[170] W. Yu, X. Lin, and W. Zhang. Towards efficient simrank computation on large networks. In C. S. Jensen, C. M. Jermaine, and X. Zhou, editors, *ICDE*, pages 601–612. IEEE Computer Society, 2013.

[171] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 283–292, New York, NY, USA, 2014. ACM.

[172] X. Yu, X. Ren, Y. Sun, Q. Gu, B. Sturt, U. Khandelwal, B. Norick, and J. Han. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM '14*, pages 283–292, 2014.

[173] P. D. H. Zadeh and M. Z. Reformat. Fuzzy semantic similarity in linked data using the owa operator. In *Fuzzy Information Processing Society (NAFIPS), 2012 Annual Meeting of the North American*, pages 1–6. IEEE, 2012.

[174] C. X. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '03, pages 10–17, New York, NY, USA, 2003. ACM.

[175] M. Zhang and N. Hurley. Avoiding monotony: Improving the diversity of recommendation lists. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, pages 123–130, New York, NY, USA, 2008. ACM.

[176] M. Zhang and N. Hurley. Avoiding monotony: Improving the diversity of recommendation lists. In *ACM RecSys '08*, pages 123–130, 2008.

[177] T. Zhou, Z. Kuscsik, J. Liu, M. Medo, J. Wakeling, and Y. Zhang. Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107:4511–4515, 2010.

[178] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 22–32, New York, NY, USA, 2005. ACM.

[179] lvaro Tejeda-Lorente, C. Porcel, E. Peis, R. Sanz, and E. Herrera-Viedma. A quality based recommender system to disseminate informa-

tion in a university digital library. *Information Sciences*, 261:52 – 69, 2014.

Appendices

Appendix A

Semantic Web technologies

In this appendix we briefly recap the basic notions of two Semantic Web standardized technologies at the base of Linked Data, i.e., RDF [48] and SPARQL [118] and we provide a quick description of DBpedia, the most relevant dataset available in the LOD cloud.

A.0.1 Resource Description Framework - RDF

The Resource Description Framework (RDF) is a general model for describing information about resources on the Web. It has been developed by the World Wide Web Consortium (W3C) in 1998 as the building block for the Semantic Web. It allows to represent Web entities and their relations as well as to attach to them a machine understandable and processable meaning (semantics) that can be further exploited to perform automatic reasoning tasks able to infer new knowledge from the explicitly stated one. Thanks to RDF, resources are made available on the Web, enabling applications to exploit them by taking into account their meaning. Each statement about resources is modeled in the form of a triple: *subject-predicate-object*. *Subjects* and *predicates* are represented by URIs, while *objects* can be identified either by URIs or by literals (data values). As an example, the two following triples are valid RDF statements about the movie *Pulp Fiction*:

```
<http://dbpedia.org/resource/Pulp_Fiction>
<http://www.w3.org/2000/01/rdf-schema#label>
"Pulp Fiction"@en

<http://dbpedia.org/resource/Pulp_Fiction>
<http://dbpedia.org/ontology/director>
<http://dbpedia.org/resource/Quentin_Tarantino>
```

where it is stated that we may refer to the the Pulp Fiction resource with the string "Pulp Fiction" and that it was directed by Quentin Tarantino. `RDF` information representation can be formally modeled through a labeled directed graph. In fact, if we consider all the `RDF` statements (triples) as a whole, what we get is a graph, where nodes are resources connected to other resources or to literal values through predicates (the edges of the graph). `RDF` can be serialized by means of different syntaxes. The most compact is the so called `turtle` syntax that allows us to use prefixes to shorten the URIs and represent them ad CURIEs. The `turtle` version of the two triples above is:

```
@prefix dbpedia: <http://dbpedia.org/resource/>
@prefix dbpedia-owl: <http://dbpedia.org/ontology/>
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>

dbpedia:Pulp_Fiction  rdfs:label  "Pulp Fiction"@en.
dbpedia:Pulp_Fiction  dbpedia-owl:director  dbpedia:Quentin_Tarantino.
```

From an ontological point of view, an interesting built-in `RDF` predicate is `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`. It states that a resource is an instance of a class.

```
@prefix dbpedia: <http://dbpedia.org/resource/>
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
@prefix dbpedia-owl: <http://dbpedia.org/ontology/>
```

```
dbpedia:Pulp_Fiction  rdf:type  dbpedia-owl:Film.
```

The previous triple asserts that *Pulp Fiction* is an instance of the class *Film*.

DBpedia Among the RDF datasets available in the LOD cloud, DBpedia is one of the main projects. It is an effort to extract structured information from Wikipedia and make it freely accessible as RDF triples. This knowledge base currently (release 3.9) describes 4 million resources, out of which 3.22 million are classified in a consistent ontology[1]. Labels and abstracts of DBpedia resources are stored in up to 97 languages. In addition, it is highly connected to other RDF datasets of the Linked Open Data cloud thus making DBpedia a cornerstone for the entire LOD project. Compared to other hierarchies and taxonomies, DBpedia has the advantage that each entity/resource is endowed with a rich description including text-based abstracts. Another advantage compared to static hierarchies is that DBpedia evolves as Wikipedia changes. Hence, problems such as domain coverage, content freshness, machine-understandability can be addressed more easily when considering DBpedia.
Each LOD dataset, including DBpedia, can be queried by means of its SPARQL endpoint. For DBpedia, it allows anyone to ask complex queries about any topic available in Wikipedia.
Noteworthy is that most DBpedia URIs have owl:sameAs links to other resources in other LOD knowledge bases[2]. Linking pieces of data across different datasets distributed on the Web is the main aim of the Linked Data initiative. Hence, we could also leverage such links for merging the data extracted from DBpedia with other data from Freebase, LinkedMDB or Yago

[1] http://wiki.dbpedia.org/Ontology

[2] The owl:sameAs property is the ontological property specifically used to link a resource in a specific knowledge base to the equivalent resources in other knowledge bases of the LOD cloud.

for example as we show at the end of the next section.

A.0.2 Simple Protocol and RDF Query Language - SPARQL

SPARQL is the de facto query language for RDF datasets. The language reflects the graph-based nature of the underlying data model. Indeed, graph-matching is its query mechanism. A basic SPARQL query is of the form:

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?c ?l
WHERE {
  dbpedia:Pulp_Fiction  dcterms:subject  ?c.
  ?c  rdfs:label ?l.
}
```

where we ask for all possible values that can be assigned to the variables ?l and ?c in order to match the graph pattern expressed in the WHERE clause. The graph pattern is represented as a set of triples where, usually, at least one of the three elements is a variable. In the previous example we have ?c as a variable for the first triple and ?c and ?l for the second one. The syntax to represent the triples follows the same rules as for RDF. SPARQL has other syntactic elements in common with RDF, such as the prefix declaration. Indeed, URIs can be represented either explicitly or in their CURIE form. Analogously to SQL, SPARQL offers different aggregation functions such as COUNT, SUM, MIN, MAX, AVG, GROUP CONCAT, SAMPLE, nested queries, negation, etc..

An interesting feature of SPARQL is the availability of filtering constraints in the representation of a graph pattern. As a way of example, the previous

query can be modified by filtering only labels represented in English.

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dcterms: <http://purl.org/dc/terms/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?c ?l
WHERE {
  dbpedia:Pulp_Fiction dcterms:subject ?c.
  ?c rdfs:label ?l.
  FILTER (langMatches(lang(?etichetta), "en"))
}
```

Other filtering constraints can be defined by comparing the value a variable is bound to, e.g., `FILTER (?var1 != ?var2)` or by applying string functions as in `FILTER (regex(?label,"ne.","i"))` where we filter the labels matching the regular expression `ne.` regardless of the letter case (case insensitive). Besides the `SELECT` query form, `SPARQL` allows the user to pose other kind of queries via `DESCRIBE`, `ASK` and `CONSTRUCT`. By `DESCRIBE` the system returns all the triples involving a particular entity. For instance, if we want to retrieve all the triples related to actors starring in both The Matrix Revolutions and Memento:

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

DESCRIBE ?actor{
  dbpedia:The_Matrix dbpedia-owl:starring ?actor.
  dbpedia:Memento_(film) dbpedia-owl:starring ?actor.
}
```

The `ASK` query form returns a Boolean answer stating if the requested subgraph is represented within the dataset.

```
PREFIX dbpedia: <http://dbpedia.org/resource/>
PREFIX dbpedia-owl: <http://dbpedia.org/ontology/>

ASK{
  dbpedia:The_Matrix dbpedia-owl:starring
  dbpedia:Carrie-Anne_Moss.
}
```

Appendix B

Further results - Chapter 3

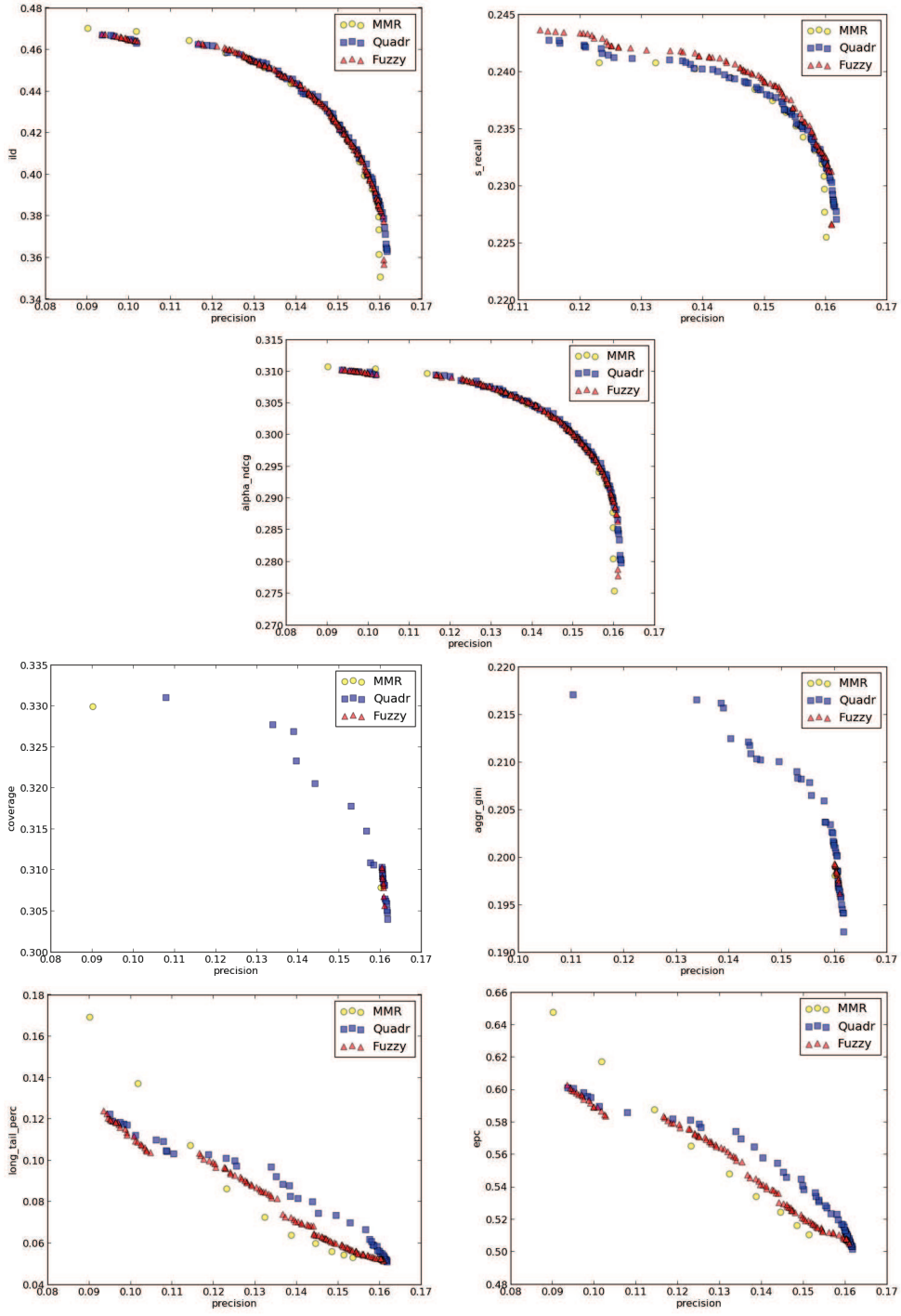


Figure B.1: Pareto Frontiers for `Movielens` Dataset, using *BPRMF* and `MMR`

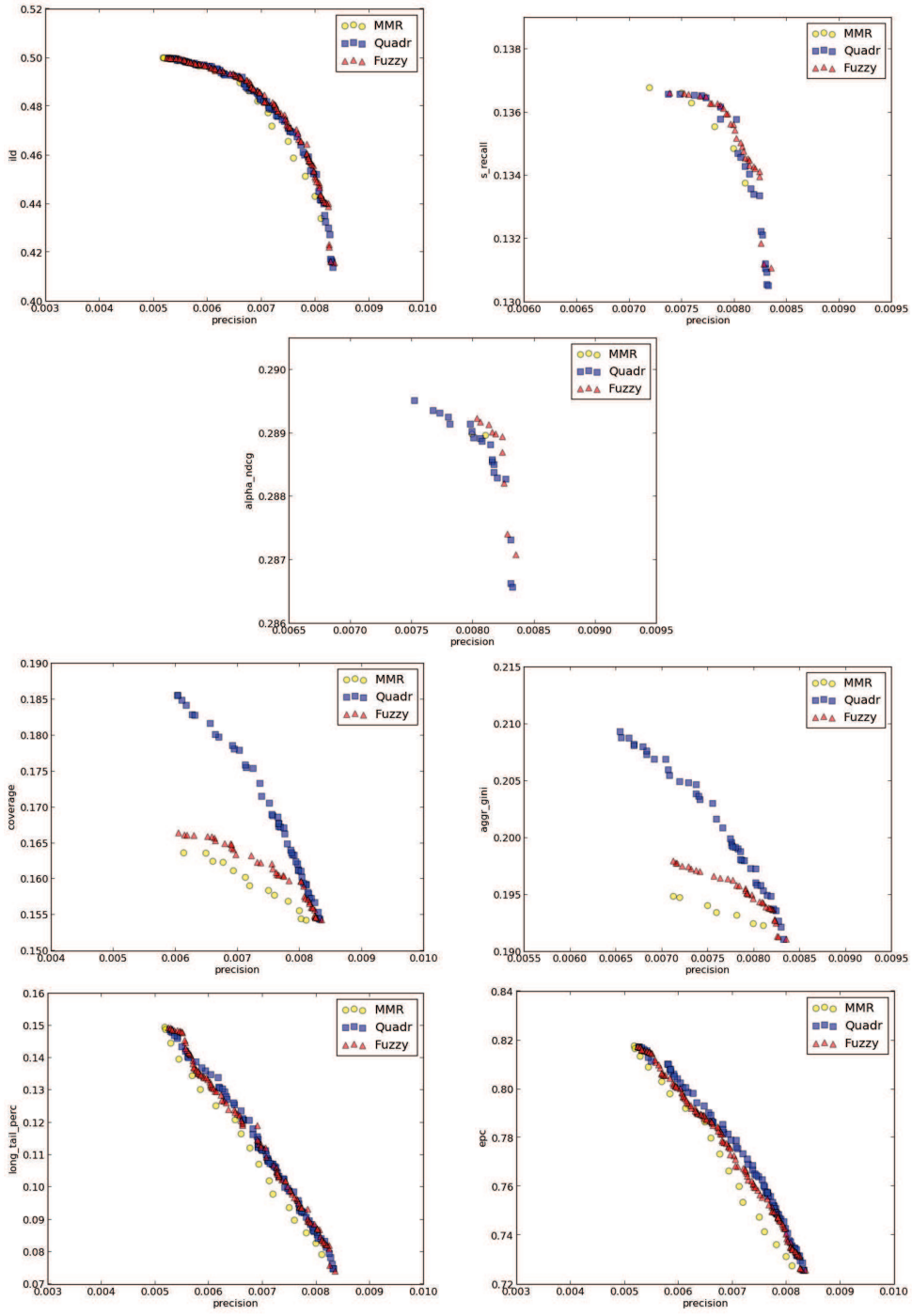


Figure B.2: Pareto Frontiers for `LibraryThing` Dataset, using *BPRMF* and `MMR`

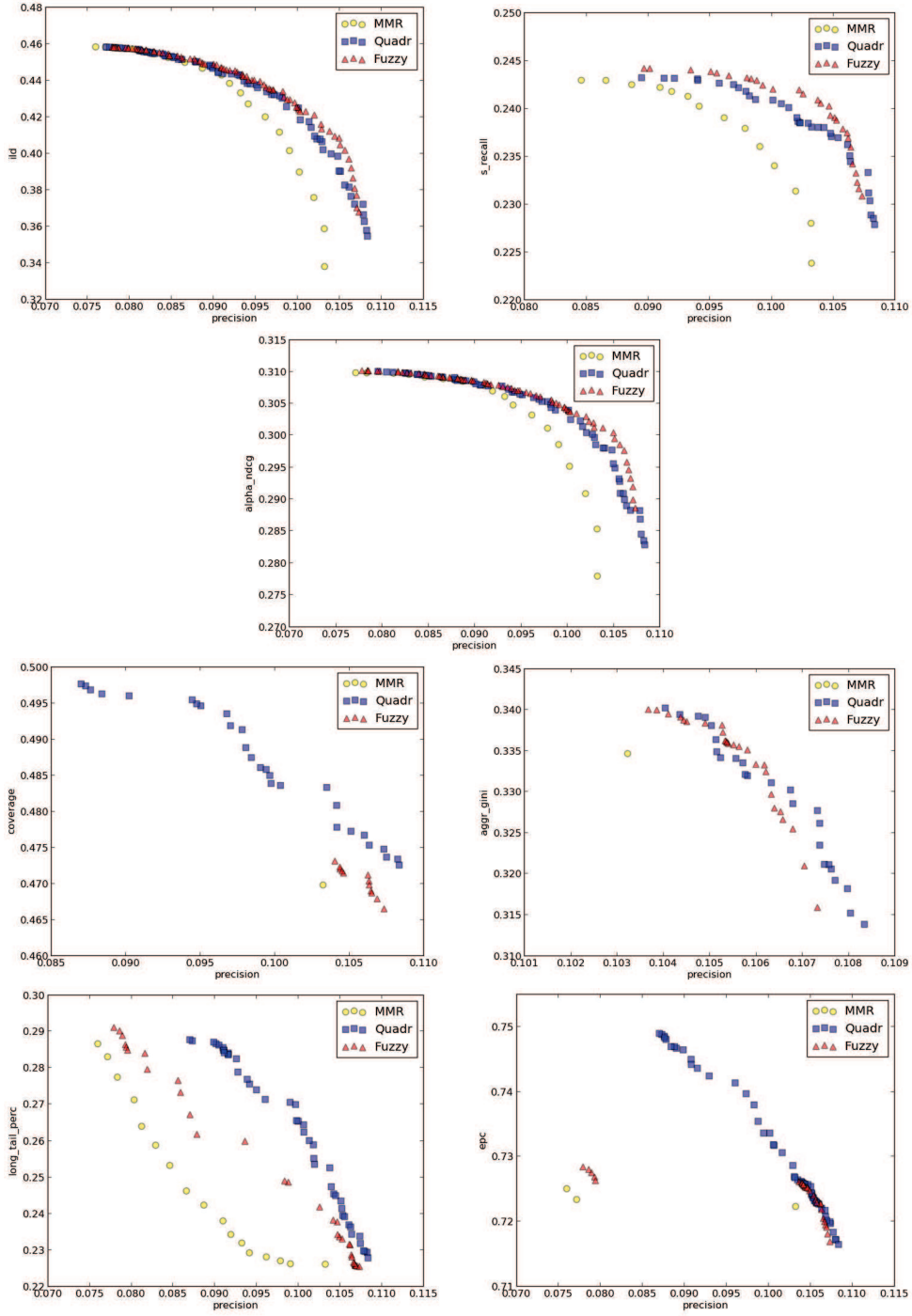


Figure B.3: Pareto Frontiers for `Movielens` Dataset, using *ItemKNN* and `MMR`

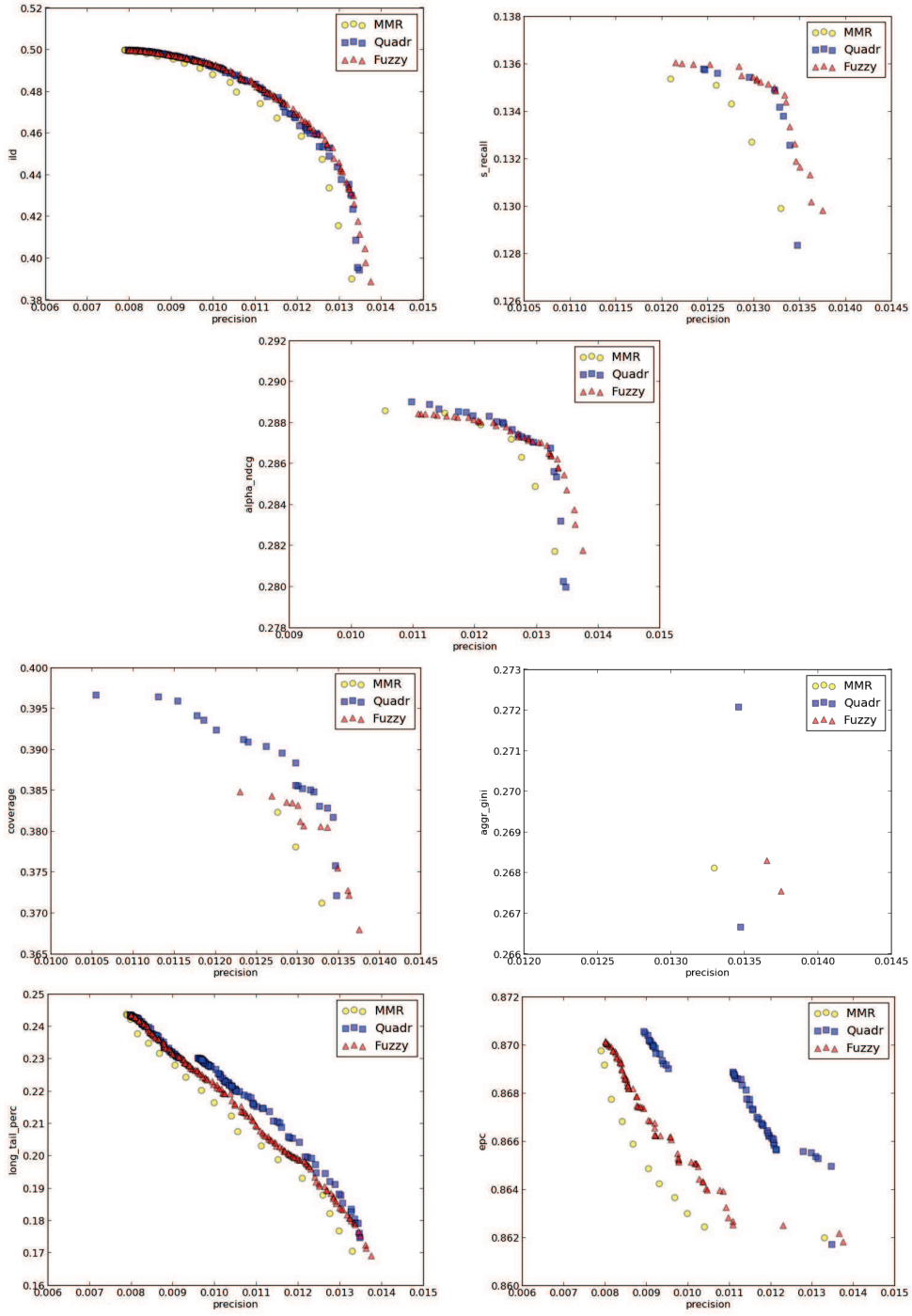


Figure B.4: Pareto Frontiers for `LibraryThing` Dataset, using *ItemKNN* and `MMR`

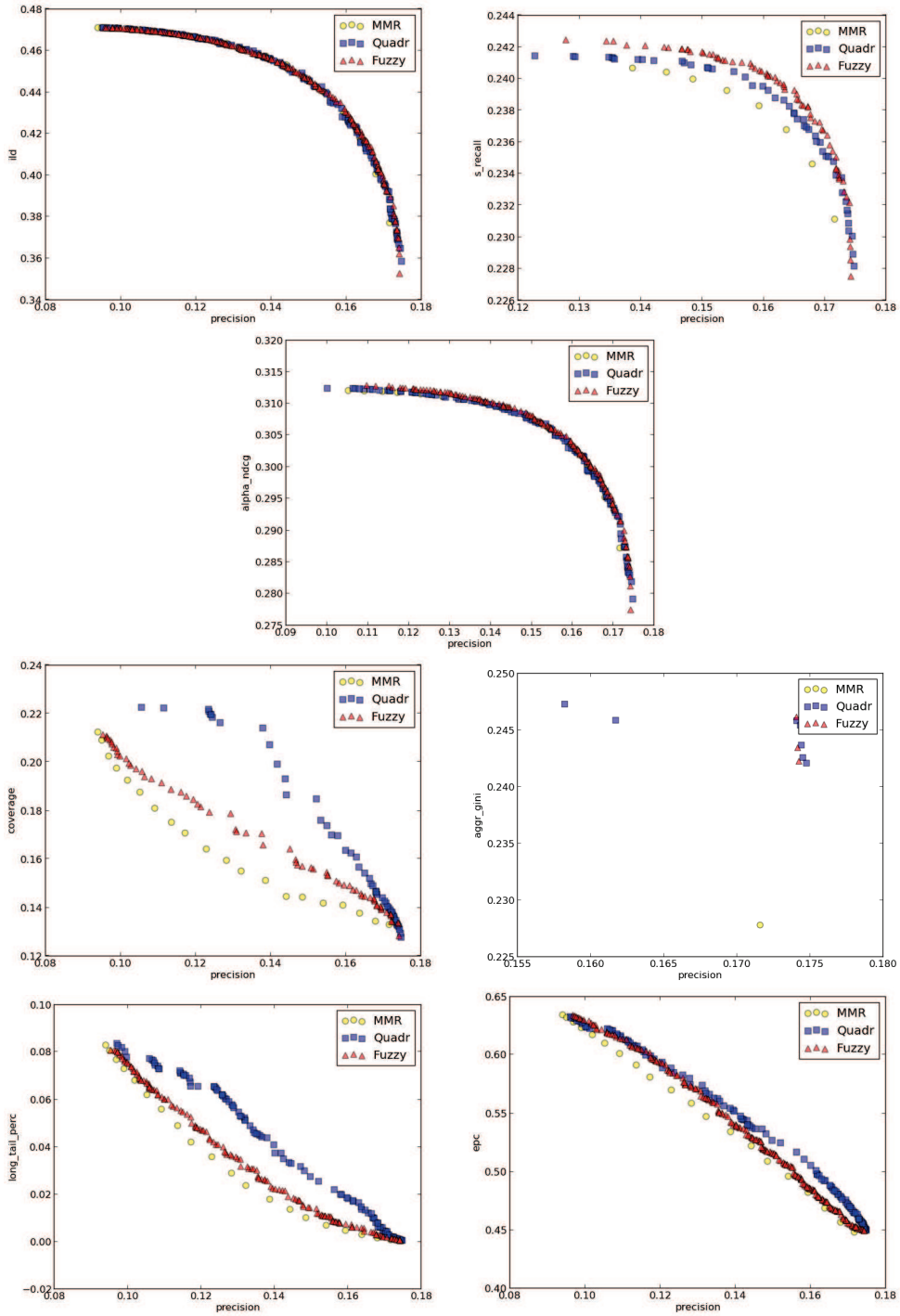


Figure B.5: Pareto Frontiers for `Movielens` Dataset, using *WRMF* and `MMR`

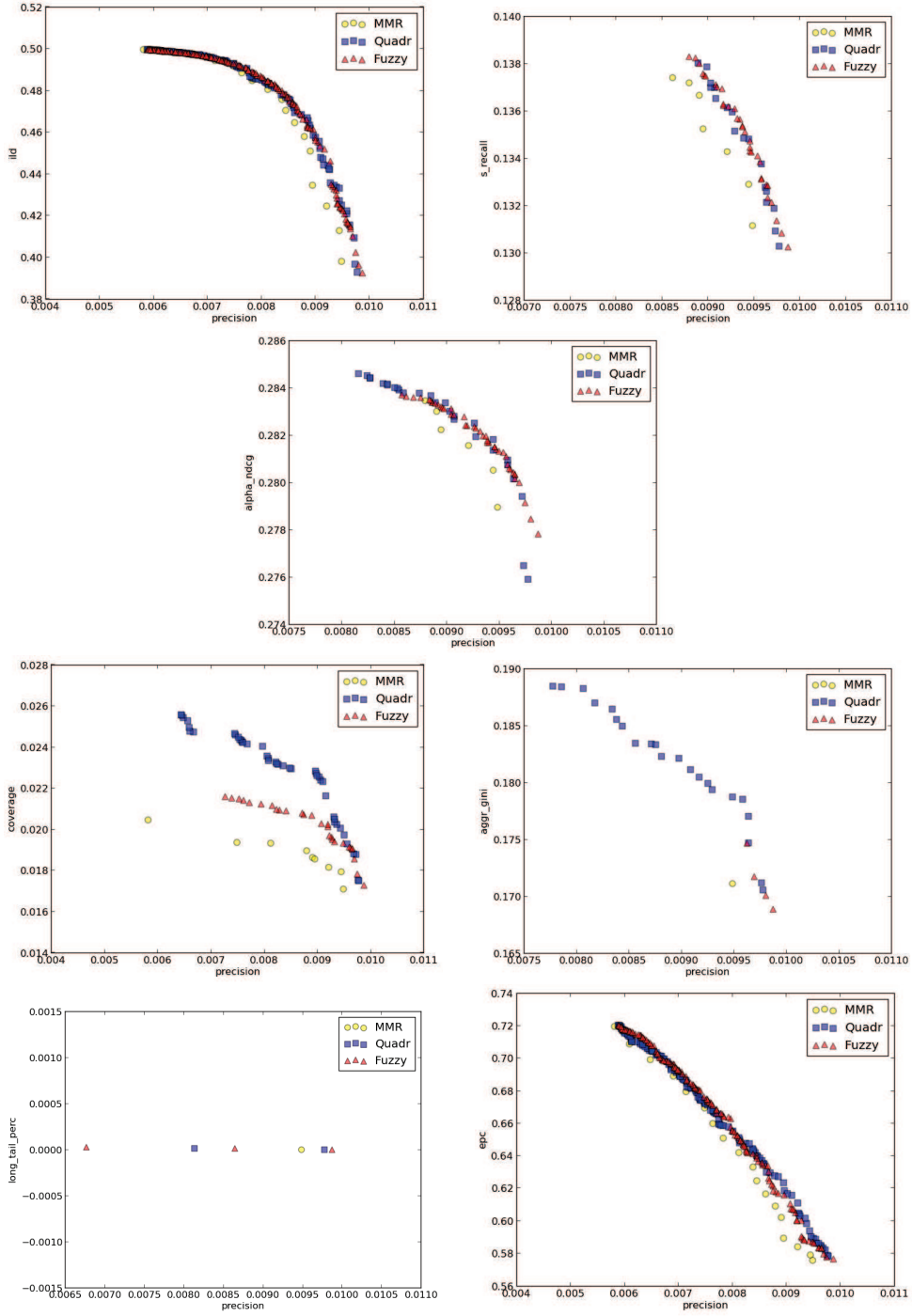


Figure B.6: Pareto Frontiers for `LibraryThing` Dataset, using *WRMF* and `MMR`

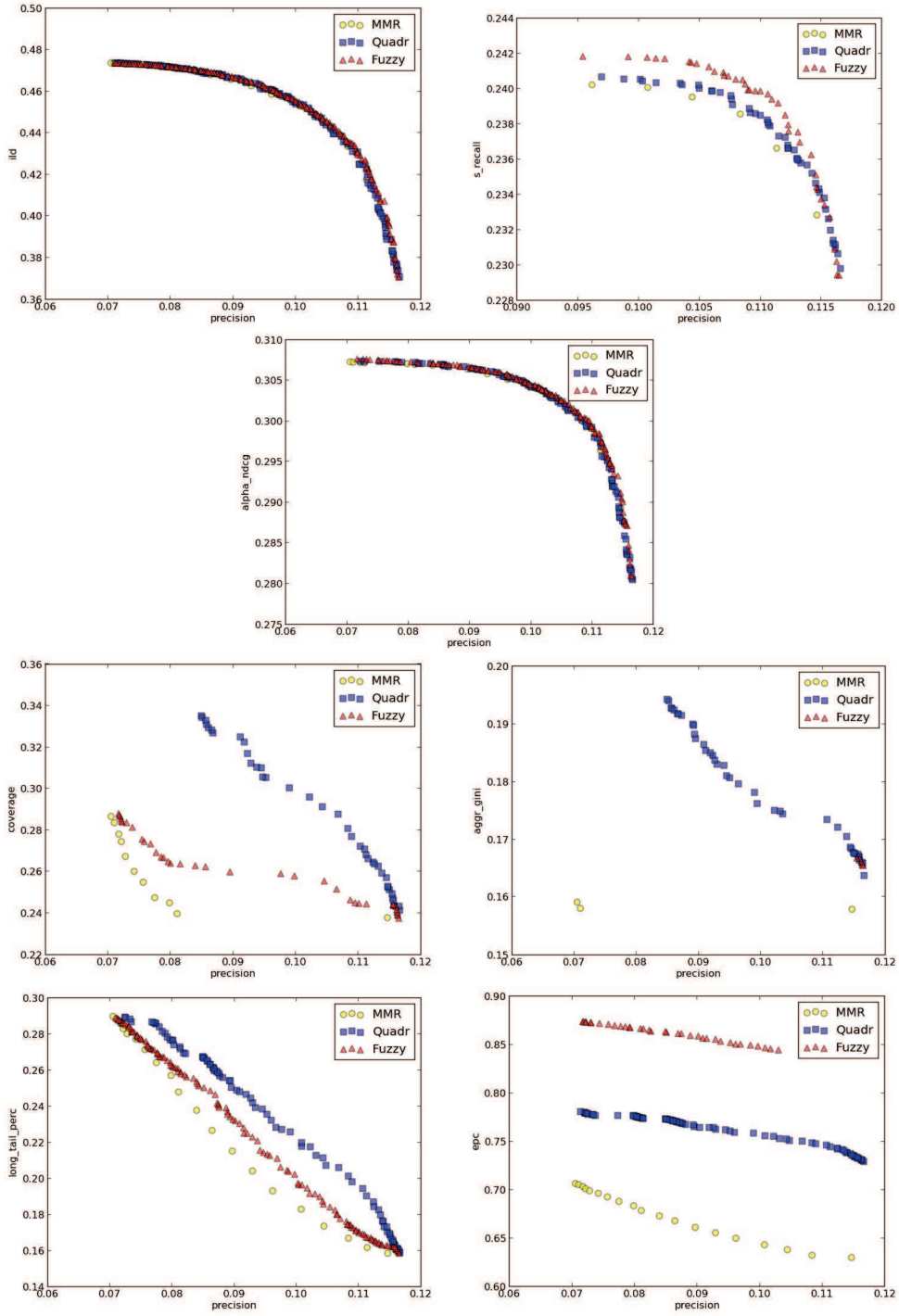


Figure B.7: Pareto Frontiers for `Movielens` Dataset, using *SoftMarginRankingMF* and `MMR`

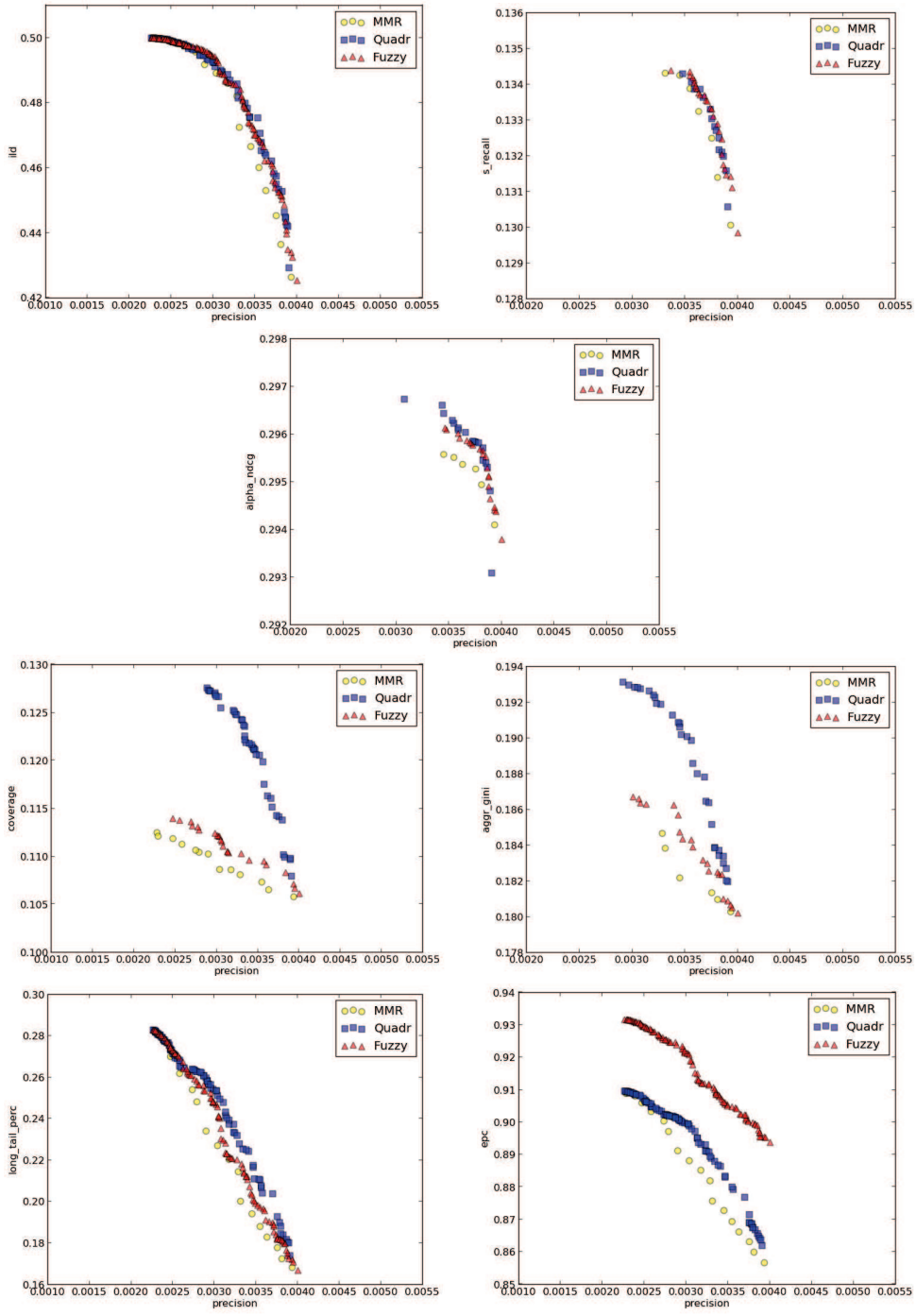


Figure B.8: Pareto Frontiers for `LibraryThing` Dataset, using *SoftMargin-RankingMF* and `MMR`

Appendix C

Further results - Chapter 4

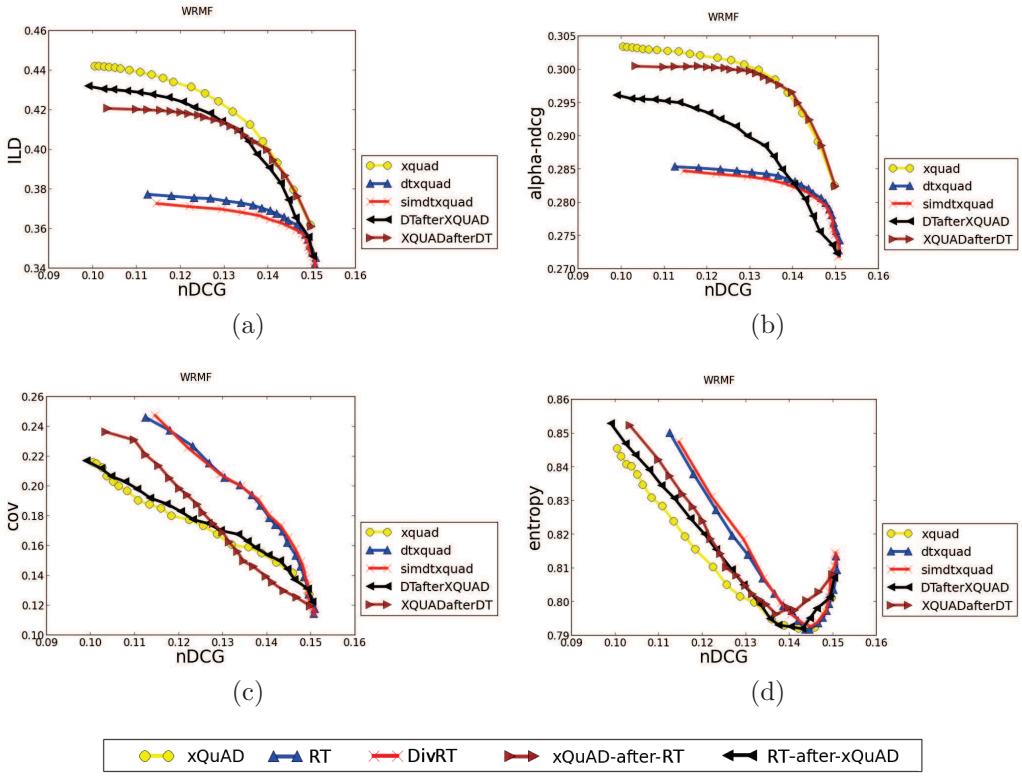


Figure C.1: Accuracy-diversity curves on MovieLens at Top-10 obtained by varying the λ parameter from 0 to 0.95 (step 0.05), using WRMF as baseline.

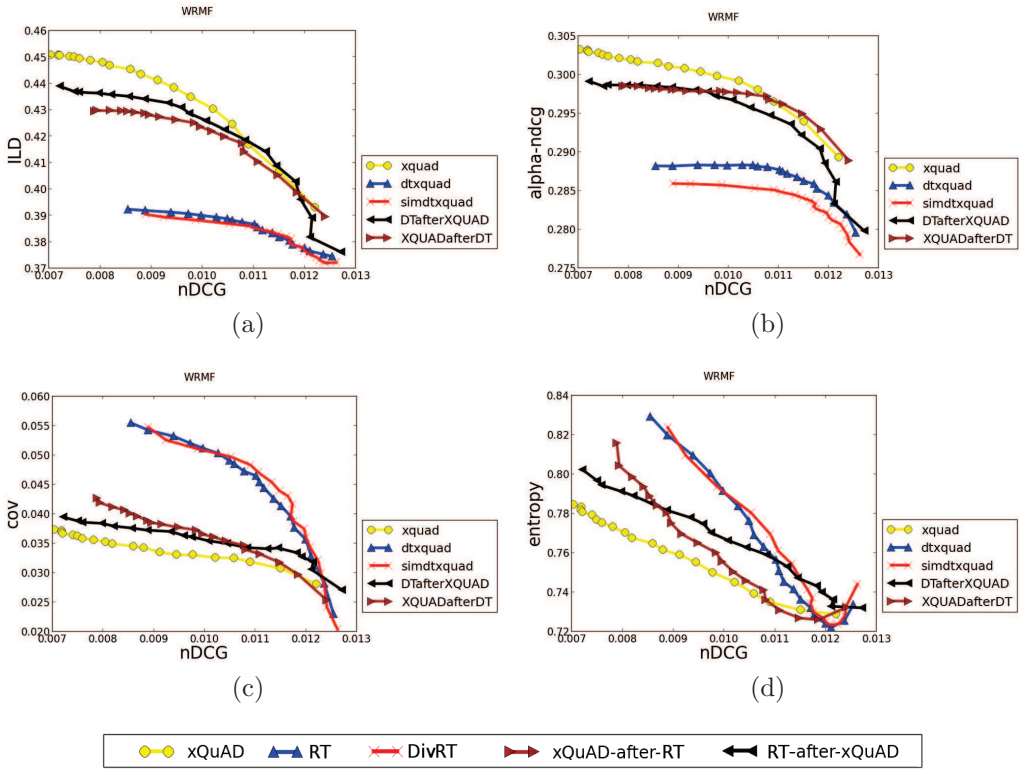


Figure C.2: Accuracy-diversity curves on LibraryThing at Top-10 obtained by varying the λ parameter from 0 to 0.95 (step 0.05), using WRMF as baseline.

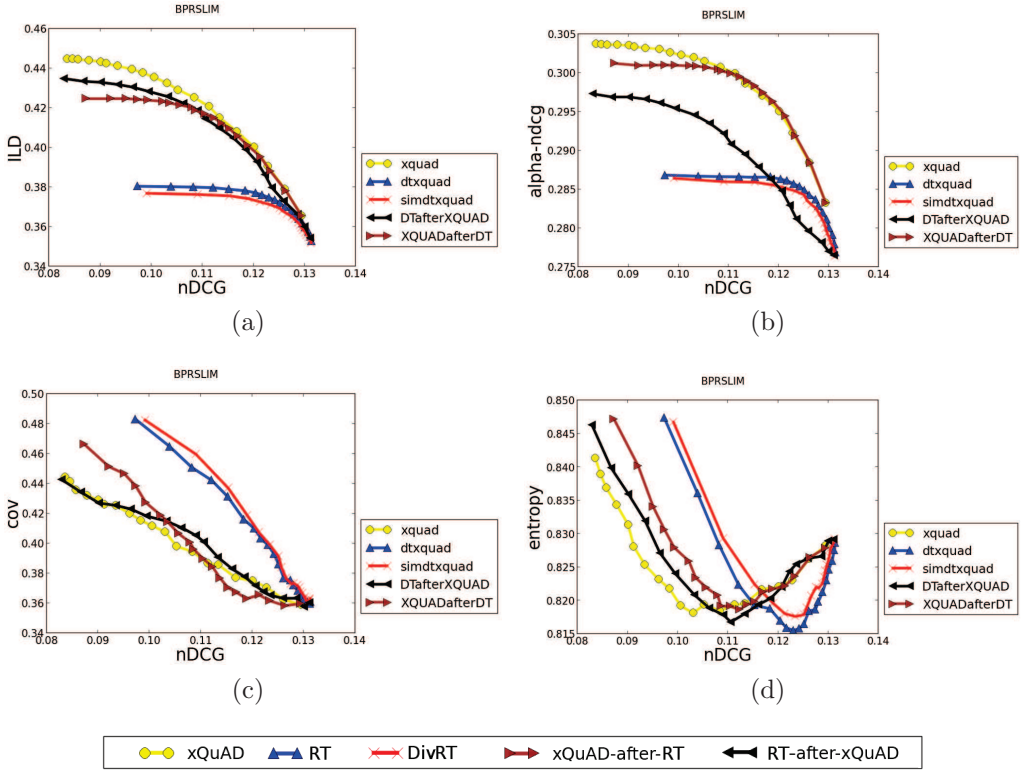


Figure C.3: Accuracy-diversity curves on MovieLens at Top-10 obtained by varying the λ parameter from 0 to 0.95 (step 0.05), using BPRSLIM as baseline.

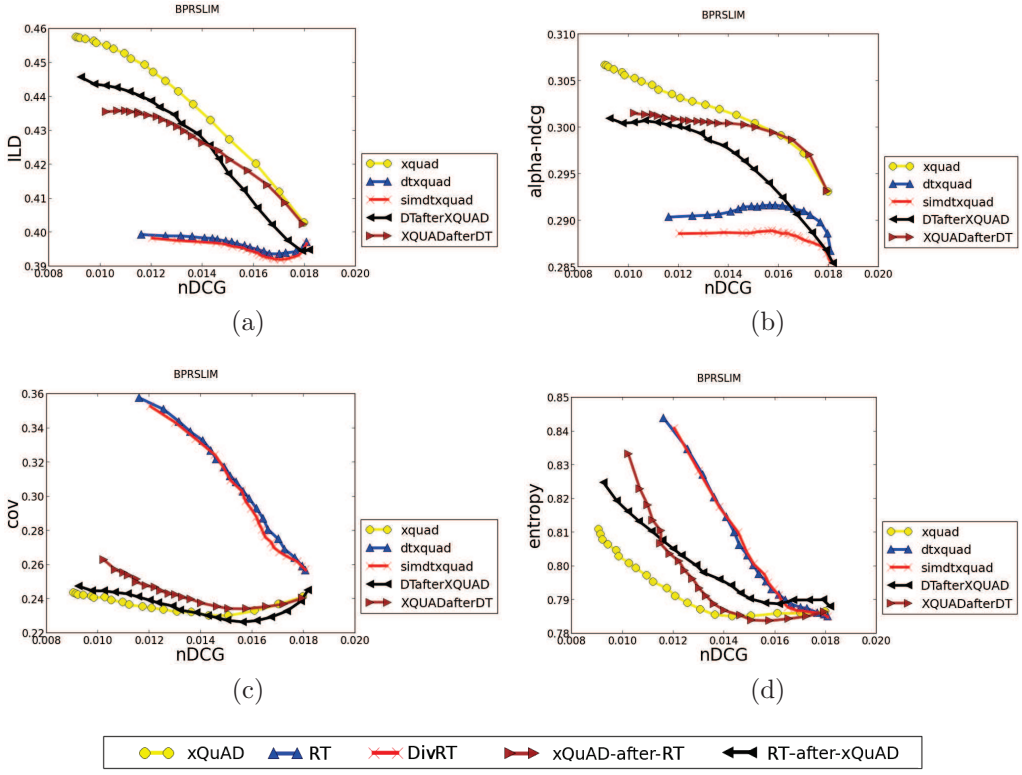


Figure C.4: Accuracy-diversity curves on LibraryThing at Top-10 obtained by varying the λ parameter from 0 to 0.95 (step 0.05), using BPRSLIM as baseline.

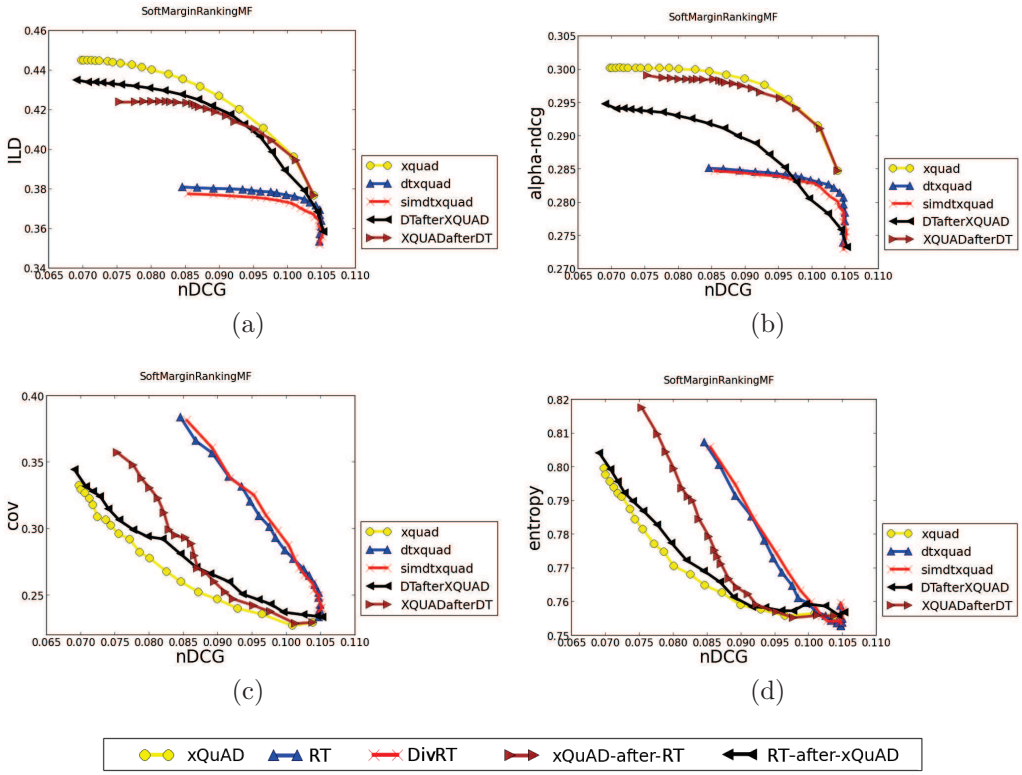


Figure C.5: Accuracy-diversity curves on MovieLens at Top-10 obtained by varying the λ parameter from 0 to 0.95 (step 0.05), using SoftMarginRankingMF as baseline.

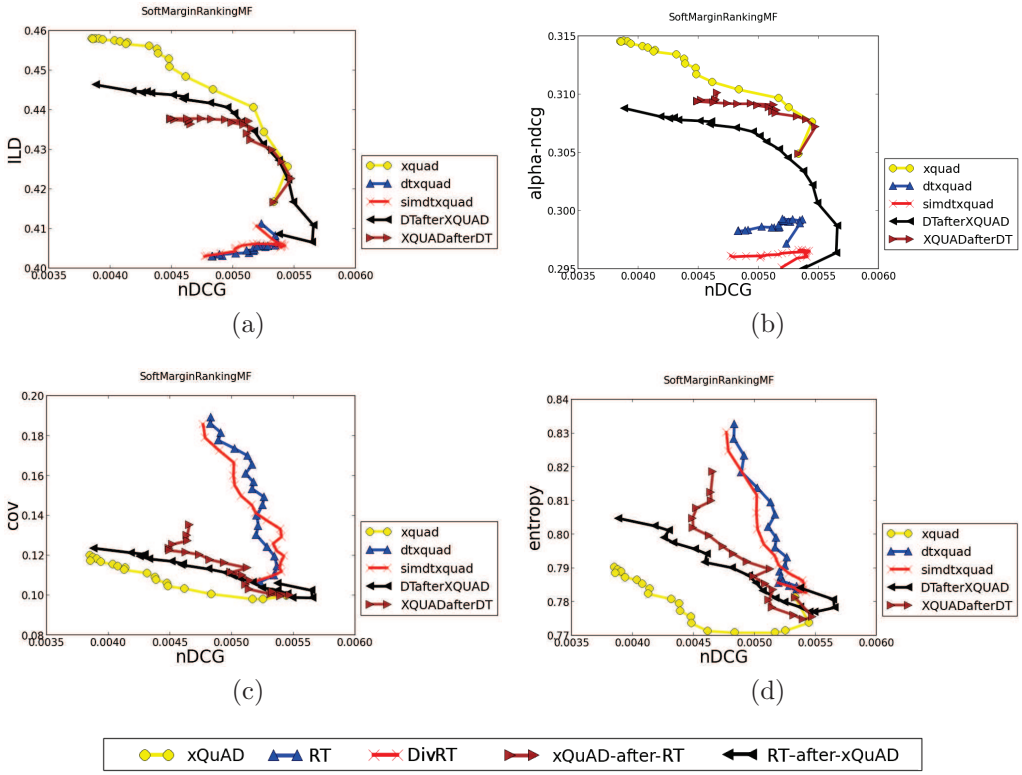


Figure C.6: Accuracy-diversity curves on LibraryThing at Top-10 obtained by varying the λ parameter from 0 to 0.95 (step 0.05), using SoftMarginRankingMF as baseline.