Department of Computer Science
University College London
University of London

Addressing the Cold Start Problem In Tag-based Recommender Systems

Valentina Zanardi

Submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
of the University College London

2010

Abstract

Folksonomies have become a powerful tool to describe, discover, search, and navigate online resources (e.g., pictures, videos, blogs) on the Social Web. Unlike taxonomies and ontologies, which impose a hierarchical categorisation on content, folksonomies directly allow end users to freely create and choose the categories (in this case, tags) that best describe a piece of information. However, the freedom afforded to users comes at a cost: as tags are defined informally, the retrieval of information becomes more challenging. Different solutions have been proposed to help users discover content in this highly dynamic setting. However, they have proved to be effective only for users who have already heavily used the system (active users) and who are interested in popular items (i.e., items tagged by many other users).

In this thesis we explore principles to help both active users and more importantly new or inactive users (*cold starters*) to find content they are interested in even when this content falls into the long tail of medium-to-low popularity items (*cold start items*). We investigate the tagging behaviour of users on content and show how the similarities between users and tags can be used to produce better recommendations. We then analyse how users create new content on social tagging websites and show how preferences of only a small portion of active users (leaders), responsible for the vast majority of the tagged content, can be used to improve the *recommender system's scalability.* We also investigate the growth of the number of users, items and tags in the system over time. We then show how this information can be used to decide whether the benefits of an update of the data structures modelling the system outweigh the corresponding cost.

In this work we formalize the ideas introduced above and we describe their implementation. To demonstrate the improvements of our proposal in recommendation efficacy and efficiency, we report the results of an extensive evaluation conducted on three different social tagging websites: CiteULike, Bibsonomy and MovieLens. Our results demonstrate that our approach achieves higher accuracy than state-of-the-art systems for cold start users and for users searching for cold start items. Moreover, while accuracy of our technique is comparable to other techniques for active users, the computational cost that it requires is much smaller. In other words our approach is more scalable and thus more suitable for large and quickly growing settings.

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Licia Capra, whose expertise, understanding, and patience, added considerably to my Phd experience. I also thank my second supervisor, Prof. Wolfgang Emmerich, who provided a contrasting viewpoint and whose insights and experience have been invaluable.

A very special thanks goes to my friends Franco Raimondi and Daniel Sykes, and to my boyfriend Enrico Scalavino that kindly assisted me in writing this thesis.

I would also like to thank all my colleagues that shared with me the Malet Place Engineering Building's 7th floor (in no particular order): Afra Mashhadi, Michalis Christodoulou, Giovanni Quattrone, Neal Lathia, Lucia Del Prete, Tamas Jambor and Will Heaven (and all the others that I forgot to mention).

I must also thank my family for the love and support they provided me through my entire life.

Contents

List of Figures

List of Tables

List of acronyms

CF	Collaborative Filtering
UBCF	User-Based Collaborative Filtering
IBCF	Item-Based Collaborative Filtering
SR	Social Ranking
CSR	Clustered Social Ranking
kNN	k-nearest neighbours
SVD	Singular Value Decomposition
PCA	Principal Component Analysis
Pop	Popularity-based recommender system
CFUT	User-based Collaborative Filtering with similarity computed with Tag usage
CFUI	User-based Collaborative Filtering with similarity computed with tagged Items
FR	FolkRank

Chapter 1

Introduction

1.1 Background

The birth and proliferation of interactive, social and customizable online resources has transformed users from passive consumers to active producers of content. This has exponentially increased the amount of available information, from videos on sites like YouTube and MySpace to pictures on Flickr, music on Last.fm, blogs on Blogger and so on. This trend has been further fostered by the enormous success of new-generation mobile devices (e.g., iPhones, Android-powered devices, Blackberry) that allow users to create and share content almost anywhere and anytime. Statistics, referring back to March 2005 [Kuchinskas, 2005], report that the online photo sharing website Flickr was comprising 775,000 registered users, hosting 19.5 million photos and was growing at 30% every month. Delicious, one of the most popular social bookmarking websites, was hosting more than 5,000,000 users and adding more than 55, 000 new posts every day [Baker, 2008].

This content is no longer categorised according to pre-defined taxonomies (or ontologies). Rather, a new trend called social (or folksonomic) tagging has emerged and has quickly become the most popular way to describe content. Unlike taxonomies, which impose a hierarchical categorisation on content, folksonomies directly allow end users to freely create and choose the tags that best describe a piece of information (a picture, a blog entry, a video clip, etc.). However, this freedom comes at a cost: since tags are defined informally and change continuously out of any control, finding content of interest has become a main challenge. The number of synonyms, homonyms, polysemous words, as well as the inevitable heterogeneity of users, make searches all the more difficult.

The problem of suggesting relevant content to users according to their interests has been widely investigated in rating-based environments, where interests are clearly expressed as numerical ratings. In these scenarios, User-Based Collaborative Filtering (UBCF) [Breese

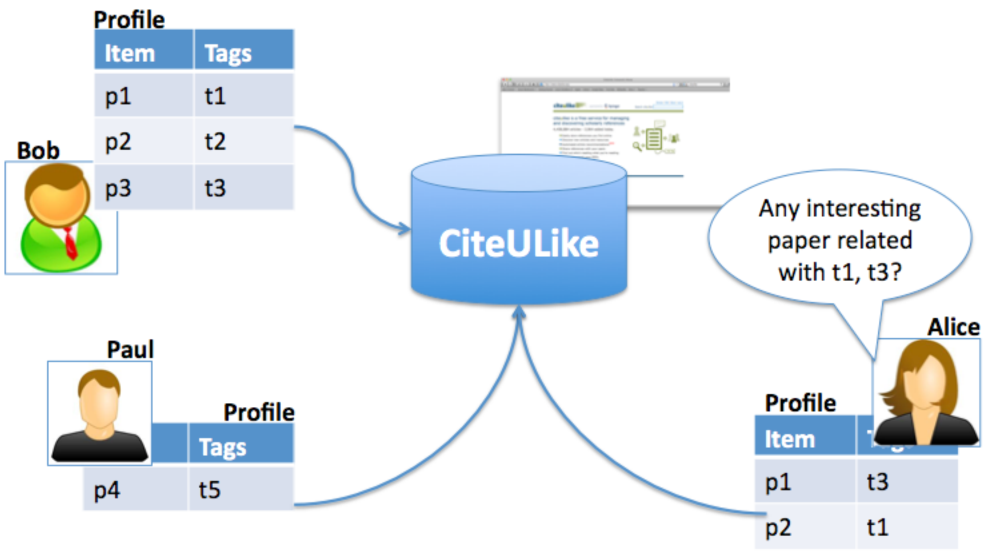

Figure 1.1: An example tagging scenario

et al., 1998] has established itself as the principal means of recommending items. The traditional UBCF approach first identifies like-minded individuals (i.e., those users who rated the same set of items with similar values). The collected ratings are then combined to predict a personalised ranked list which is recommended to each user.

The model proposed by UBCF could also be applied in tag-based scenarios. However, any such application must consider that tags introduce a new third dimension in the standard two-dimensional relationship between users and items. Moreover, while in rating-based scenarios users usually rate all content they know something about, regardless of the opinion they have about it (i.e., they rate also items they do not like), in tag-based environments users tend to tag only content they are interested in. This behavior makes the process of learning interests even more challenging. To see why, we provide a more detailed example next.

1.2 Motivating Scenario

Consider a community of researchers interested in storing, organizing and exchanging relevant scientific papers. Rather than using the existing rigid ACM classification system[1], users could freely describe papers with tags from spoken language. Websites offering this kind of service already exist. An example is the CiteULike website (Figure 1.1). CiteULike allows users to describe scientific references with freely chosen tags which produce

[1]http://www.acm.org/about/class/

a folksonomy of academic interests. For each registered user, the system builds a user profile based on all the papers the user tagged and the set of tags she used. Formally, a folksonomy is defined as a tuple $F := (U, T, P, Y)$ [Hotho et al., 2006], where:

- U, T and P are finite sets, whose elements are called users, tags and resources (papers as in the considered example), respectively. Referring to Figure 1.1, $U := (Alice, Bob, Paul)$, $T := (t_1, t_2, t_3, t_5)$ and $P := (p_1, p_2, p_3, p_4)$.
- Y is a ternary relation between the sets, i.e., $Y \subseteq U \times T \times R$, where each tuple (u, t, r) is called *bookmark*.

Consider a registered user Alice, logging on the CiteULike website and interested in receiving recommendations about papers she may deem relevant. We analyze the impact of applying two different state-of-the-art recommendation strategies to accomplish the described task: **User-Based** and **Item-Based Collaborative Filtering** (UBCF and IBCF respectively).

1.2.1 User-Based Collaborative Filtering in Tag-Based Settings

To recommend interesting content to Alice, UBCF processes all users' profiles and identifies a subset of users who have similar interests to Alice. In the following, we will consider users to have similar interests if they either tag a same subset of items or use a same subset of tags. Their opinions are then used to produce recommendations under the assumption that users who shared interests at a given time t_1 will still do so at any time $t_2 > t_1$.

However, identifying users with similar interests in our tag-based scenario is a more challenging process than in traditional rating-based environments, as users express their preferences by associating items with tags rather than using numerical ratings. We can compare the performance of the UBCF algorithm on both rating-based and tag-based environments by considering two simple examples depicted in Table 1.1 and Table 1.2. For the sake of simplicity, we assume that every user has tagged each item with one tag only.

We first focus on the rating-based environment depicted in Table 1.1. User Alice shares similar opinions on items p_1 and p_2 with user Bob. Both users in fact rated the two items p_1 and p_2 with value 5. In this scenario, a simple UBCF algorithm would identify Bob as a possible recommender for Alice and would suggest item p_3 to her, just because Bob has a positive opinion about it.

Consider now a similar situation but in the tag-based environment depicted in Table 1.2. In this scenario, Bob could be identified as a possible recommender for Alice, since both users tagged items p_1 and p_2. However, this is not a viable conclusion as it does not consider that Alice and Bob used different tags to describe p_1 and p_2. Bob described item

	Item			
User	p_1	p_2	p_3	p_4
Bob	5	5	4	
Alice	5	5		
Paul				4

Table 1.1: Rating pattern over existing items

	Item			
User	p_1	p_2	p_3	p_4
Bob	t_1	t_2	t_3	
Alice	t_3	t_1		
Paul				t_5

Table 1.2: Tag pattern over existing items

p_1 with tag t_1 and item p_2 with tag t_2, while Alice described item p_1 with tag t_3 and item p_2 with tag t_1. This could imply that users are interested in the same items but from different perspectives and that they are unlikely to share interests.

Similarly, identifying Bob as a possible recommender by looking at commonly used tags (both users used tags t_1 and t_3) is not a viable solution either, as it does not consider that the tags were used on different items. For example t_1 might be a polysemic word and might have two different meaning when it is associated with one paper or another.

Problem statement: applying traditional UBCF techniques in tag-based environments implies flattening the tri-dimensional relationship between users, items and tags to two dimensions and consequently discarding potentially useful information.

In tag-based environments, a novel recommendation technique capable of leveraging the tri-dimensional relationship between users, items and tags is thus called for.

Let us focus now on user Paul, who has recently joined the system. After logging in on the website, Paul expresses his interest for item p_4 by tagging it with tag t_5. Since Paul was the only user in the system to tag the new item p_4 and to use the new tag t_5, a traditional CF approach would fail in selecting a set of possible recommenders for him. This problem is well-known in the literature as the *user cold start problem* and appears to be aggravated in tag-based scenarios, as user preferences are not expressed by unambiguous ratings from a finite and discrete numerical domain. Conversely, tags generate wide folksonomies often in the order of thousands of keywords (e.g., CiteULike has roughtly 130,000 tags). If tag

t_5 were a synonym of tag t_2, Bob could be identified as a possible recommender for Paul (as they would share one tag in common) and items p_1, p_2 and p_3 could be considered by the recommendation engine.

Problem statement: In both rating- and tag-based environments, predictions about what items a target user may be interested in cannot be computed if there is no or little overlap between her rated/tagged items and those rated/tagged by the community. Moreover, in tag-based environments, if a target user uses new or unpopular tags, or tags which are synonym to the ones commonly used by the community, predictions about what items she may be interested in cannot be computed as no suitable recommenders can be selected.

In tag-based environments, a novel technique capable of producing recommendations even for new or inactive users, and for users expressing their preferences with unpopular or synonym tags, is thus called for.

1.2.2 Item-Based Collaborative Filtering in Tag-Based Settings

An alternative technique to suggest interesting content to Alice is *Item-Based Collaborative Filtering* (IBCF). IBCF processes Alice's profile information and identifies a subset of items which are similar to those she tagged and which can be recommended to her. The underpinning assumption is that users' interests do not change over time. In other words if Alice was interested in a set of items S_1 at time t_1, at any time $t_2 > t_1$ she is likely to enjoy a set of items S_2 similar to S_1. However, identifying similarities between items in tag-based environments is much more challenging than in a traditional rating-based ones.

We can compare the performance of the IBCF algorithm on both rating-based and tag-based environments by considering the examples depicted in Table 1.3 and Table 1.4. We first focus on the rating-based environment depicted in Table 1.3. Alice expresses a preference for item p_1, rating it with value 4. In this scenario, a simple IBCF algorithm would identify item p_2 as similar to p_1, since both users Bob and Paul share similar opinions on them (they both rated p_1 and p_2 with value 5). The system would thus suggest item p_2 to Alice, given the fact that it appears to be similar to her rated item p_1.

Consider now the tag-based scenario depicted in Table 1.4. In this case, identifying item p_2 as similar to item p_1 simply because it was tagged by both Bob and Paul is not a viable solution as it does not consider that the tags used were different. Bob described item p_1 with tag t_1 and item p_2 with tag t_3, while Paul described item p_1 with tag t_2 and item p_2 with tag t_1.

Let us focus now on item p_3, belonging to a narrow and unpopular research field and

	User			
Item	**Bob**	**Paul**	**Steve**	**Alice**
p_1	5	5		4
p_2	5	5		
p_3			5	

Table 1.3: Rating pattern over existing items

	User			
Item	**Bob**	**Paul**	**Steve**	**Alice**
p_1	t_1	t_2		t_1
p_2	t_3	t_1		
p_3			t_5	

Table 1.4: Rating pattern over existing items

thus tagged by user Steve only. Since p_3 cannot be identified as similar to any other existing item, it will never be recommended to any user. This problem is well-known in the literature as the *item cold start problem* and appears to be aggravated in tag-based scenarios, as user preferences are not expressed by unambiguous ratings. For example, if tag t_5 were a synonym of tag t_1, p_3 could be considered similar to p_1 and p_2 and would thus be considered by the recommendation engine.

Problem statement: In rating- and tag-based environments, new items cannot be recommended to users if they are not similar to any other existing item. Moreover, in tag-based environments not only new items but also items tagged with new or unpopular tags, or with tags which are synonyms of existing ones, cannot be recommended to users as they cannot be identified as similar to any of the other existing items.

In tag-based environments, a novel technique capable of producing recommendations even for items described by unpopular or synonym tags is thus called for.

The *user* and *item cold start problems* are dominant in recommender systems where new users join and new items are uploaded everyday. As for March 2007, the Citeulike website had 33,000 registered users and was gaining new registrations at the rate of 100 per day. At that time there were 505,402 items in the database tagged 1,676,130 times using 130,548 distinct tags [Emamy and Cameron, 2007]. These numbers have been continuously growing since then. In this scenario, an effective search and recommender system must be capable of recommending items even when little information is available about user interests and items.

Furthermore, as also emphasized by Anderson in his book "The Long Tail" [Anderson, 2006], the almost endless variety of items which can be suggested to users and the increased demand for medium-to-low popularity items make traditional recommendation strategies unsuitable to meet preferences of users interested in niche products. In the rest of this work we will refer to items tagged by few users as "medium-to-low popularity" or "non-mainstream" items. An effective search and recommender system must be capable of including non-mainstream items in order to satisfy users with atypical preferences.

1.2.3 Scalability

In addition to the *cold start problems*, tag-based recommender systems have to face bigger scalability problems than traditional rating-based ones. This is because almost everyone can produce content and the number of users and items grows at a very high rate. Furthermore, we are dealing with a tri-dimensional problem, where not only users and items but also tags grow at a very high rate. In recent years, the research community has been very active to devise algorithms capable of producing ever more accurate recommendations, but partly left aside the scalability problem. The Netflix prize competition[2], promising an award of 1 million dollars to whoever could improve the accuracy of the Netflix system by 10%, has further fostered research towards accurate systems, while neglecting scalability. Indeed, the proposed solutions took into account several hundred different algorithms and trillions of different variables and their complexity put scalability at risk. Even Neil Hunt, Netflix's chief product officer, has admitted the advantages of more precise recommendations may be outweighed by the cost of the additional computation required.

Problem statement: There is a conflicting tradeoff between system accuracy and system scalability.

A scalable technique capable of producing accurate recommendations at low computational cost is thus called for.

Recommender system techniques specifically developed for tag-based scenarios need to be lightweight not only during the online recommendation process. The offline process of building and periodically updating the pre-computed data structures which are required to produce online recommendations must be very efficient too. Given the high computational cost, traditional strategies perform data updates at fixed time intervals (for example, every week or every fortnight). As a consequence, the system periodically relies on stale data structures that might hinder recommendations.

Problem statement: There is a conflicting tradeoff between the benefits of frequent data updates and their corresponding costs.

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

A novel strategy to perform data updates only when the benefits outweigh the corresponding costs is thus called for.

1.3 Thesis Contributions

In this thesis we focus on scenarios where items are characterised by descriptive tags. Numerical ratings are not available and users preferences can only be inferred from tagged items. The goal of this thesis is to develop new, effective and scalable recommendation techniques capable of suggesting interesting items even when little information is available about user interests. In addition, the set of recommendations must include unpopular items, if these are relevant to the target user. An overview of the main contributions of this thesis is given in the following.

1.3.1 Accurate Recommendations for Cold Start Users and Items

We start our investigation by analyzing the key features of CiteULike, a typical tag-based social tagging website. The results of this analysis point out that the considered dataset is rather sparse. Most of the users belonging to the system have seldom used it and most of the tags and items have been used/tagged by a small subset of users only. This data sparsity causes traditional recommender systems to be particularly sensitive to both the *user* and *item cold start problems*. In this thesis, we propose a new technique capable of suggesting even unpopular items both to active and new or inactive users. To achieve this goal, we project the existing tri-dimensional relationship between users, items and tags onto two two-dimensional relationships, users-tags and tags-items respectively. We use the former to compute similarities between users and to find suitable recommenders, and the latter to compute similarities between tags and to discover potentially useful tags which can bring the attention of users on interesting items.

We develop an algorithm called Social Ranking (SR) that exploits these similarity metrics to effectively suggest relevant items to all users of the system.

1.3.2 Scalable Recommendations

As the results of our previous investigation point out, and as also confirmed by current statistics, the vast majority of items is tagged by a rather small proportion of users (*leaders*), while other users (*followers*) mainly browse them. Moreover, leaders tend to share interests with a rather small group of other users only, suggesting that they have clearly defined interests that map to a small subset of all the existing items. In this thesis, we propose a technique that identifies the leaders and clusters them in different domains of interests on the basis of their tagging activity. We develop an algorithm called Clustered Social Ranking (CSR) that exploits the interests of this small core set of selected leaders to provide effective recommendations for both active and new users. The small but meaningful information subset exploited by CSR ensures that the system scales well with the data.

1.3.3 Adaptive Temporal Evaluation

Since all recommendation algorithms rely on pre-computed data structures which must be kept up-to-date, studying how data grows is crucial. We have thus performed an extensive analysis of the growth of tag-based datasets to understand the rate at which new users, tags and items appear and the impact they have on accuracy, should the underlying data structure not be kept up-to-date. Results suggest that data growth is not consistent between users, tags and items and, more importantly, differs between different datasets. Based on this observation, we define a new adaptive methodology capable of deciding when the data structure must be updated depending on the growth of data and thus the expected accuracy loss. This allows us to perform system updates only when the expected benefits outweigh the cost.

1.3.4 Evaluation of Results

To prove the effectiveness and efficiency of the proposed techniques, we have implemented both SR and CSR and evaluated them on three different social tagging websites, each dealing with different media resources: 1) CiteULike organizes scientific references; 2) Bibsonomy organizes both scientific references and general URLs and 3) MovieLens organizes movies.

We evaluate the performance of our proposed techniques against state-of-the-art recommendation algorithms and show that they produce more effective recommendations for *cold start users* and *cold start items* across all datasets. For active users, our techniques ensure comparable recommendations while achieving significantly better scalability.

1.4 Thesis Outline

The remainder of this thesis is structured as follow:

Chapter 2 introduces the approach devised to enhance recommendation accuracy for *cold start users* and *items*. We first present state-of-the-art recommendation techniques and then discuss their limitations. We then analyze the CiteULike website to illustrate the problem we are dealing with. Finally, we describe Social Ranking, the technique we propose to tackle the *cold start problems* in our the scenario.

Chapter 3 offers a thorough evaluation of SR in terms of precision and recall on three different social bookmarking websites, namely CiteULike, Bibsonomy and Movielens. Performance achieved by SR are compared against four common benchmarks.

Chapter 4 introduces the approach devised to enhance the scalability of the recommendation process. We first analyze a number of state-of-the-art recommendation approaches and their computational complexity. We then describe the key properties of our target scenario. We then present Clustered Social Ranking, a technique we have devised to achieve precision and recall comparable to state-of-the-art approaches, while significantly improving the scalability of the system.

Chapter 5 offers a thorough evaluation of CSR both in terms of precision and recall and in terms of scalability. Performance achieved by CSR are again compared against four state-of -the-art benchmarks.

Chapter 6 describes our temporal investigation of the growth of social tagging websites. We first describe and evaluate the results of our experiments, to show how data growth affects the performance of our recommender system. Finally, we describe a temporal adaptive technique to predict the accuracy loss that using stale data would cause, and thus to decide whether the benefits of a data update are worth its cost.

Chapter 7 summarises and evaluates the contribution of our work and explores possible directions for future research.

1.5 Publications Related to This Thesis

The following publications are related to this thesis:

- Valentina Zanardi and Licia Capra. A Scalable Tag-based Recommender System for New Users of the Social Web. Paper under review for the *22nd International Conference on Database and Expert System Applications (DEXA 2011).*

- Valentina Zanardi and Licia Capra. Dynamic Updating of Online Recommender Systems via Feed-Forward Controllers. In *Proceedings of the 6th Intl. Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011). Waikiki, Honolulu, Hawaii, USA, May 2011.*

- Valentina Zanardi and Licia Capra. Social Ranking: Uncovering Relevant Content Using Tag-based Recommender Systems. In *Proceedings of the 2nd ACM International Conference on Recommender Systems (RecSys). Lausanne, Switzerland, October 2008.*

- Valentina Zanardi and Licia Capra. Social Ranking: Finding Relevant Content in Web 2.0. In *Proceedings of ECAI 2008, Workshop on Recommender Systems. Patras, Greece, July 2008.*
- Daniele Quercia, Licia Capra and Valentina Zanardi. Selecting Trustworthy Content Using Tags. Invited paper at *SECRYPT, Special Session on Trust in Pervasive Systems and Networks. Porto, Portugal. July 2008.*

Chapter 2

Cold Start Users and Items

The first contribution of this thesis is the development of a new methodology to deal with the *cold start user* and *item problems*. In this chapter, we first introduce state-of-the-art recommender system techniques for the social web (Section 2.1) and highlight their limitations. We then analyse the problem domain in detail (Section 2.2) and present the key principles we exploit to address it (Section 2.3). Finally, we describe Social Ranking (SR), the innovative technique we have developed (Section 2.4).

2.1 Related Work

2.1.1 Recommendation Algorithms for Rating-Based Scenarios

The amount of information available online has long exceeded the capacity of individual users to process it. This caused a strong interest in research fields and technologies that could help manage information overload. Recommender systems have thus been created to suggest relevant data to potentially interested users. Dating back to 1992, Tapestry [Goldberg et al., 1992] can be considered one of the first recommender systems. Tapestry was specifically designed to improve the functionalities of simple mailing lists. To ensure that all users interested in an e-mail received the message, Tapestry allowed users to put annotations on e-mail messages so that recipients could filter them through specific commands. Although very simple, the recommendation methodology proposed by Tapestry was not automated. Users had to explicitly query the system to receive the e-mails they were interested in. The birth of a wide number of rating websites, where users can explicitly rate their preferred items, further pushed research in this field. In these scenarios, *Collaborative Filtering* (CF) [Su and Khoshgoftaar, 2009] has been considered as the state-of-the-art of recommender techniques. The fundamental assumption of CF is that similarities between users' interests are persistent. If users have shown similar interests in the past (e.g., they rated items similarly), they are likely to do so in the future too. CF techniques store users' preferences and identify users with similar rating behaviours. Whenever a target user asks for recommendations, ratings expressed by similar users are used to produce a list of potentially interesting items to suggest. In the following sections, we review a set of CF approaches we consider relevant for this thesis. Following a traditional classification, CF algorithms can be partitioned into three main categories: *Memory-Based*, *Model-Based* and *Mixed or Hybrid Approaches.*

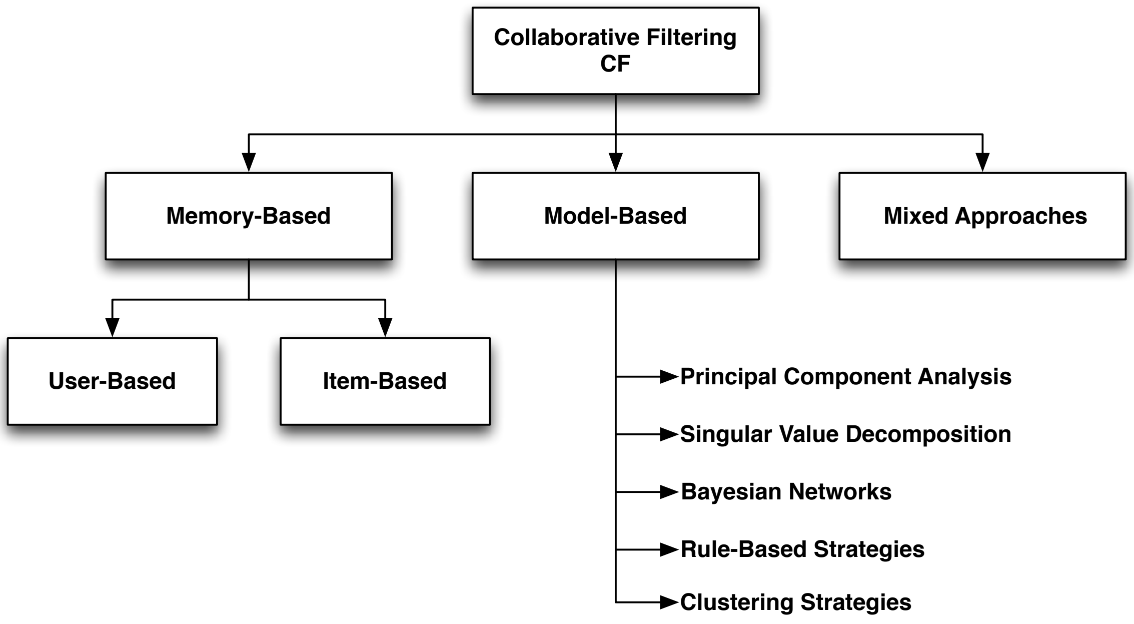

Figure 2.1: Classification of recommendation techniques for rating-based scenarios

Memory-Based Collaborative Filtering algorithms rely on all user-item information to produce recommendations. These algorithms first store all the users' ratings into memory and combine them to predict users' interests on items. The two existing variants of memory-based recommendation algorithms are both based on the k-nearest neighbour algorithm (kNN) [Aha et al., 1991]: *User-Based Collaborative Filtering* and *Item-Based Collaborative Filtering.*

User-Based Collaborative Filtering algorithms (UBCF) [Herlocker et al., 1999] produce recommendations for a target user u_j by first identifying a subset N_{u_j} of existing users expressing interests similar to u_j and using them as recommenders. In other words, for each user pair (u_j, u_l) the similarity $sim(u_j, u_l)$ is calculated and N_{u_j} is built accordingly. In the following we will refer to N_{u_j} as the neighbourhood of u_j. The interest of u_j for each item can be predicted by combining her neighbours' preferences. A recommendation list containing the top k most interesting items is then given in output. Different strategies can be used to compute the subset N_{u_j} and to combine the preferences of users in N_{u_j}. Each user is first modeled by a profile vector, containing for each rated item o_i the corresponding numerical rating. Then, one of the many similarity measures analyzed by Breese et al. [Breese et al., 1998] can be used to compare profiles. One of the most popular is perhaps cosine similarity that measures the cosine of the angle between two user profiles vectors u_j and u_l:

$$sim(u_j, u_l) = \sum_{i=1}^{|O|} \frac{r_{u_j,o_i} * r_{u_l,o_i}}{\sqrt{\sum_{i=1}^{|O_j|} r^2_{u_j,o_i}} \sqrt{\sum_{i=1}^{|O_l|} r^2_{u_l,o_i}}} \tag{2.1}$$

where O is the set of items rated by both u_j and u_l, O_j and O_l are the set of items rated by u_j and u_l respectively and r_{u_j,o_i} is the rating u_j gave to item o_i. GroupLens [Resnick et al., 1994, Konstan et al., 1997] proposes instead a UBCF algorithm for Usenet news articles that uses the Pearson correlation to compute N_{u_j}. This statistical coefficient measures the linear dependence between two user profiles u_j and u_l:

$$sim(u_j, u_l) = \frac{\sum_{i=1}^{|O|}(r_{u_j,o_i} - \overline{r}_{u_j})(r_{u_l,o_i} - \overline{r}_{u_l})}{\sqrt{\sum_{i=1}^{|O_j|}(r_{u_j,o_i} - \overline{r}_{u_j})^2 \sum_{i=1}^{|O_l|}(r_{u_l,o_i} - \overline{r}_{u_l})^2}} \tag{2.2}$$

where $\overline{r}_{u_j}$ refers to the mean rating of u_j. The most traditional approach to compute the predicted rating r'_{u_j,o_i} of item o_i for the target user u_j (used also by GroupLens) is to calculate the weighted average of the ratings of the users in N_{u_j}:

$$r'_{u_j,o_i} = \frac{\sum_{\forall u_l \in N_{u_j}} (r_{u_l,o_i} * sim(u_j, u_l))}{\sum_{\forall u_l \in N_{u_j}} sim(u_j, u_l)} \tag{2.3}$$

Item-Based collaborative filtering algorithms (IBCF) [Linden et al., 2003] follow the same general principle as UBCF. The two approaches differ in the usage of the stored rating data. While UBCF considers it as a collection of users who have rated items, IBCF considers it as a collection of items that have been rated by users. IBCF does not identify a neighborhood of similar users but explores the relationships between items first. In particular, each item is first modeled as a vector containing, for each user, the corresponding numerical rating. IBCF then scans the items $o_i \in R_{u_j}$ the target user has rated in the past, and finds the set R'_{u_j} of the k most similar items not rated yet. Similarities between items can be computed using the same measures used by UBCF for users. The predicted rating r'_{u_j,o_i} of item $o_i \in R'_{u_j}$ for the target user u_j is usually computed as the weighted average similarity with all items o_q in R_{u_j}:

$$r'_{u_j,o_i} = \frac{\sum_{\forall o_q \in R_{u_j}} (sim(o_i, o_q) * r(u_i, o_q))}{\sum_{\forall o_q \in R_{u_j}} sim(o_i, o_q)} \tag{2.4}$$

Traditionally, since available datasets tend to have many more users than items, IBCF is often preferred over UBCF. Sarwar et al. [Sarwar et al., 2001] analyze and evaluate different implementations of IBCF, comparing performance when the Pearson correlation or cosine similarity are adopted to generate R'. They also compare performance when using weighted average or regression to obtain the final predictions. Regression is similar to the weighted sum method but, rather than directly using the ratings of similar items, it uses an approximation of the ratings based on the following regression model:

$$ra_{u_j,o_i} = \alpha r_{u_j,o_i} + \beta + \epsilon \tag{2.5}$$

where ra_{u_j,o_i} is the approximated rating, α and β are the regression model parameters and ϵ is the error of the regression model. Experimental results computed over the MovieLens dataset demonstrate that adopting the cosine similarity measure leads to improved performance. Moreover, the authors demonstrate that using regression leads to improved performance only in sparse datasets, while weighted average appears to be more suitable as the density of the dataset increases.

The commercial success of memory-based Collaborative Filtering techniques led to the creation of a number of more effective variants. Several studies have focused on improving the methodology used when generating neighborhoods. The Ringo Music Recommender [Shardanand and Maes, 1995] improved the performance of UBCF by limiting neighborhood's membership only to those users whose similarity was greater than a fixed threshold. However, even if such technique could provide more accurate recommendations, the higher threshold sensibly reduced the number of items Ringo was able to generate predictions for. The Bellcore Video Recommender [Hill et al., 1995] combined opinions of a randomly selected neighborhood in order to generate recommendations instead. Other studies have focused on creating better user models integrating different information sources such as contextual [Anand and Mobasher, 2007], bibliografic [Krulwich, 1997], or a domain ontology [Haase et al., 2004, Anand et al., 2007]. Finally, other methods propose to fuse together UBCF and IBCF. Wang et al. [Wang et al., 2006] use a probabilistic method to compute the predicted rating r'_{u_j,o_i} on the basis of: 1) the ratings given to the same item by similar users, 2) the ratings given to similar items by the same user and 3) the ratings given to similar items by similar users. Note that this list does not intend to be exhaustive but gives an idea of the success and evolution of memory-based Collaborative Filtering algorithms. Despite their wide-spread use, these systems still suffer from *data sparsity* and from the *scalability* problem.

Recommender systems are generally used to recommend items from large sets, so even the most active users usually express preferences for a rather restricted number of selected items only. In this scenario, a recommender system using memory-based CF techniques may be unable to find like-minded users and thus to provide any useful recommendation. This problem, referred to as the *data sparsity* problem, occurs whenever users show non-overlapping profiles (e.g., they rated different items) and makes the similarity evaluation impossible or not reliable. Furthermore, memory-based CF strategies suffer from excessive complexity whenever the number of users and items grows too large. Whenever a user asks for recommendations, these strategies need to compare all existing users to identify the target user's neighborhood and to combine neighbours' preferences to produce a top-k recommendation list. Given the size of the dataset these strategies deal with, memory-based recommender systems suffer from serious *scalability problem.*

A different group of approaches, generally referred to as **Model-Based Collaborative Filtering algorithms**, has been proposed to ease the scalability problem. These algorithms develop a model of user ratings rather than identify a neighborhood of similar users. Recommendations are then produced simply using the developed model. Models can be built with various strategies, such as Singular Value Decomposition (SVD) or Principal Component Analysis (PCA) [Takacs et al., 2009, Goldberg et al., 2000, Sarwar et al., 2000], rule-based strategies [Mobasher et al., 2001] and clustering strategies [Connor and Herlocker, 2001, Suryavanshi et al., 2005]. The most representative model-based techniques will be described in more detail in Section 4.1. Since these techniques exploit only the developed model to compute predictions rather than focusing on the original database of ratings, they can provide accurate recommendations while easing the scalability problem. However, these techniques are still affected by the data sparsity problem: developing a model of user ratings is challenging when data is missing.

Different techniques [Xue et al., 2005, Ungar and Foster, 1998] propose to combine the strengths of memory-based and model-based approaches together to better deal with both the scalability and the data sparsity problems. Xue et al. [Xue et al., 2005] propose to apply first a model-based technique that groups users into clusters based on their rating patterns. Given a target user, her cluster is first found and the opinions of users in that cluster are combined to compute the final prediction as in memory-based techniques. Similarly, in [Ungar and Foster, 1998] the authors propose a repeated clustering technique that separately groups both users and items. The final predicted rating r'_{u_j,o_i} is then computed by first selecting the cluster of u_j and of item o_i and by using the corresponding information. Despite their efficacy, model-based techniques for rating-based environments have been developed to model a strictly two-dimensional scenario were only users and items are involved in the recommendation process.

All previously described techniques share the common goal of recommending items that users will probably be interested in. In particular, these algorithms could be reused and adapted to accomplish the same task in tag-based environment where users' tastes are not clearly expressed as numerical ratings but as freely chosen tags associated with items. However, folksonomies are so large and dynamic that traditional web recommendation techniques are no longer effective [Heymann et al., 2008]. Novel techniques to help users find relevant content are thus called for.

2.1.2 Learning and Exploiting the Hidden Semantic of Tags

A number of studies [Sen et al., 2006, Halpin et al., 2007, Cattuto et al., 2007, Golder and Huberman, 2006] have been conducted to investigate users' tagging behavior and to derive a model of vocabulary evolution in social tagging communities. These studies show that the tagging activity of users does not solely depend on their personal preferences but is also influenced by the general tagging behavior of the entire community. For this reason, the set of applied tags remain limited, as tags are being reused. In other words, users with similar interests (members of the same community) are likely to employ tags on items in similar ways. Relationships between users and the set of tags they selected can therefore be usefully leveraged for different tasks.

One stream of research focused on inferring the semantic relationship between tags, starting from an analysis of how users employ them. Heymann et al. [Heymann and Garcia-Molina, 2006] tried to build a navigable hierarchical taxonomy of tags, purely starting from tag usage. Shen et al. [Shen and Wu, 2005] propose to describe the tagging behavior of users as a probabilistic model and to automatically derive relationships between tags. Yeung et al. [Yeung et al., 2007] propose a simple technique to disambiguate tags, i.e., to interpret the correct meaning of a tag based on an analysis of the relationships between users, tags and items. Capocci et al. [Capocci and Caldarelli, 2008] study tag co-occurrence by modeling the network of users, tags and items as a tri-partite graph. Once again, the aim is to discover semantic relationships between tags, starting from information about how users associate them with items.

Other approaches [Hassan-Montero and Herrero-Solana, 2006, Kaser and Lemire, 2007] have built upon these studies to develop *tag cloud visualization algorithms.* A tag cloud is a set of the most popular tags, or the most recently used tags, usually displayed with different font sizes according to their popularity. In a tag cloud, users can click on tags and obtain an ordered list of items described by that tag. These techniques, which focus only on the number of times each tag has been used, are aimed at providing a simple visual interface which can be useful to browse information.

2.1.3 Recommendation Algorithms for Tag Based Environment

Many researcher have started to investigate how to exploit the previously learned tags relationships to develop recommendation algorithms that assist users finding relevant items within folksonomies.

A few approaches propose to associate tags with a fixed ontology of concepts to better answer users' queries. In [Passant, 2007] and [Pan et al., 2009], authors propose to link each query tag to a specific class of an ontology to expand the original query with a set of useful tags. Doing so, authors provide a method to address the limitations of simple recommendations techniques based on common tags' matching. The first limitation is caused by *tag variations*. If related items are described by synonyms, the system cannot recognise them as similar and fails in recommending them all. The second limitation is caused by *tag ambiguity*. If two tags have the same spelling but two different meanings, the system is not aware that the items the two tags are associated with are different. However, the high overhead to create and maintain the ontology makes these approaches feasible only in specific scenarios where the set of tags which can be used is limited or where the cost of maintaining such ontology can be afforded (e.g., in medical environments).

Other approaches have proposed methods inspired by traditional recommender system techniques for rating-based scenarios, where the tagging activity of users is analysed to find relationships between tags. Some approaches have focused on mixed scenarios where both numerical ratings and tags are available. For example, Tso et al. [Tso-Sutter et al., 2008] propose an extension of a traditional UBCF algorithm where similarity between users depends not only on co-rated items, but also on commonly used tags. Similarly, in [Nakamoto et al., 2007] the authors propose to analyse only commonly used tags to calculate similarity between users. The predicted score for each item is then computed as in traditional collaborative filtering solutions, depending on the ratings of similar users. Other approaches have been developed to target pure folksonomic settings where numerical ratings are not available and user preferences can only be inferred by analyzing users' tagging activity. In [Ji et al., 2007] and [Bogers and van den Bosh, 2009] the authors propose two variants of the traditional UBCF and IBCF algorithms for folksonomic scenarios. The user-based version of the algorithm first evaluates users similarity based on commonly used tags and builds the target user's neighborhood. The set of items already tagged by the neighborhood is then suggested, weighted according to the similarity between the target user and the tagging user. The item-based version of the algorithm works similarly. Items' similarity is first computed based on commonly used tags and the set of items which are most similar to those previously tagged by the target user is suggested. Wetzker et al. [Wetzker et al., 2009] propose a probabilistic approach to item recommendation in folksonomies. By using Probabilistic Latent Semantic Analysis strategies, the authors derive a model of users' tagging behaviour. By analyzing the item-tag and the item-user co-occurrance, the authors provide two separate tagging models which are merged together using Expectation-Maximization strategies. The authors tested their approach on a large part of the Delicious dataset and found it outperforms popularity-based algorithms.

In addition, a variety of approaches have been proposed to suggest useful tags that users could use to describe their favourite items. Gemmel et al. [Gemmell et al., 2009] propose instead to use first a modified version of the standard kNN strategy to compute similarities between users based on commonly used tags. Once the target user's neighborhood has been built, all tags used by the neighborhood and associated with the target item are suggested. The relevance of each tag is then computed as a weighted average similarity between the target user and the tagging user in the neighborhood. Symeonidis et al. [Symeonidis et al., 2008] propose instead a unified framework to model the three types of entities that exist in a social tagging system: users, items and tags. This data is represented by a binary 3-dimensional matrix M_t called tensor, so that $M_t(u_j, o_i, t_k) = 1$ if user u_j tagged item o_i by using tag t_k and 0 otherwise. The authors propose to apply first a dimensionality reduction algorithm over M_t to obtain three approximated matrices that preserve only a percentage of information of the original data. These matrices are then combined together so that the original 3-dimensional matrix M_t can be reconstructed into an approximated matrix $\hat{M_t}$. Based on the values stored in $\hat{M_t}$, a set of tags can be recommended to the target user. This approach, called Tensor Reduction approach, is able to reveal the hidden associations between users, items and tags that can be used to improve the tag recommendation process. Rendle et al. [Rendle et al., 2009] propose a slightly different version of the Tensor Reduction approach, aimed at improving the data model that is used to construct the original matrix M_t. M_t is built so to include not only positive or negative examples ($M_t(u_j, o_i, t_k) = 1$ or 0 according to the users tagging activity), but also missing data ($M_t(u_j, o_i, t_k) =?$ if item o_i was not tagged at all by user u_i). By discarding missing data when applying the Tensor Reduction approach, the authors are able to outperform the standard Tensor Reduction algorithm both in terms of precision and recall. Hotho et al. [Hotho et al., 2006, Jschke et al., 2008] propopose FolkRank, a PageRank-like algorithm that employs the traditional random surfer model on the tri-partite graph of users-items-tags, producing very accurate tag recommendations in well connected networks. To be effective, FolkRank requires that every user, item and tag appear in the system at least p times so to ensure that there is a minimum amount of information shared between users. Every time a user asks for recommendations, the algorithm ranks users, tags and items according to the number of their mutual connections in the graph. It then returns the best k tags.

As for traditional rating-based recommender systems, major issues left open by state-of-the-art tag-based recommender systems are the *user* and *item cold start problems.* However, these problems are aggravated in tag-based scenarios since users' preferences are not expressed as unambiguous numerical ratings, but as freely chosen tags. When new users join the system, very little is known about their interests and predictions about what items they may be interested in are difficult to compute. Similarly, when new items are tagged only by a small subset of users, or when different tags are used to describe similar items, the system does not have enough information to produce reliable recommendations. In situations of such sparsity which are all but rare in tag-based scenarios, different techniques specifically developed to support new users and items are called for. Some researches have already moved in this direction. For example, Massa et al. [Massa and Avesani, 2007] propose to replace the old concept of similarity between users (which is not computable if users have not rated enough common items), with a new concept of trust where users explicitly state who their trusted recommenders are. Such approaches are viable only in scenarios where creating a user's social network comes at no extra cost. Other researchers [Schein et al., 2002, Leung et al., 2007, Lam et al., 2008] try to exploit metadata information concerning both users (e.g., age, gender and job) and items (e.g., genre and cast of movies), to compute recommendations when tagging/rating information is scarce. Unfortunately, in most tag-based websites metadata is not available. The applicability of these approaches is therefore limited.

In the next sections we provide a detailed analysis of the problem space on which we focus and underline its characteristics to build a solution that can specifically address the *user* and *item cold start problems.*

2.2 Analysis of the Problem Space

To analyse the characteristics of our scenario, we chose a typical social tagging website: CiteULike (http://www.citeulike.org). CiteULike is a social tagging website for the sharing of scientific references. Similar to the cataloging of web pages within Delicious and of photographs within Flickr, CiteULike allows scientists to organize their libraries with freely chosen tags which constitute a folksonomy of academic interests. The CiteULike dataset is freely available and contains a list of user-item-tag tuples, indicating what article (item) has been tagged by whom and with which tags. Note that since user-item pairs can be associated with several tags, in the following we will refer to *bookmark* as the set of user-item-tag triples with the same user-item pair. The downloaded archive contains bookmarks made between November 2004 and November 2009. We preprocess the dataset to get rid of noise by removing all non-alphanumeric tags, following the same methodology proposed by the organisers of the ECML PKDD Discovery Challenge 2009[1]. The so-pruned archive contains 41,246 users who have tagged 1,254,406 papers with 210,385 distinct tags.

We analysed the dataset in terms of *users' activity*, *papers' popularity*, and *tags' usage.* Note that we verified the results reported below by also analysing two other datasets, namely Bibsonomy and MovieLens, and we concluded we can consider them to be valid for general social tagging websites.

User Activity. We studied how many papers were tagged on average by each user in the system (Figure 2.2) and found that 66% of the users tagged less than 10 papers (low activity), 20% tagged between 10 and 50 papers (medium activity) and the

[1]http://www.kde.cs.uni-kassel.de/ws/dc09/

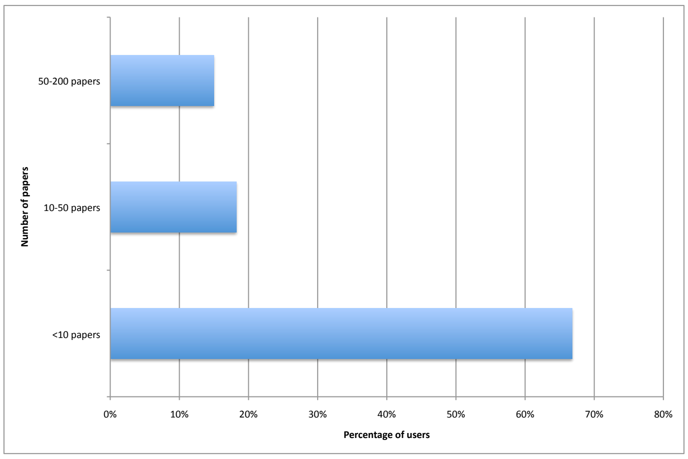

Figure 2.2: User activity on papers

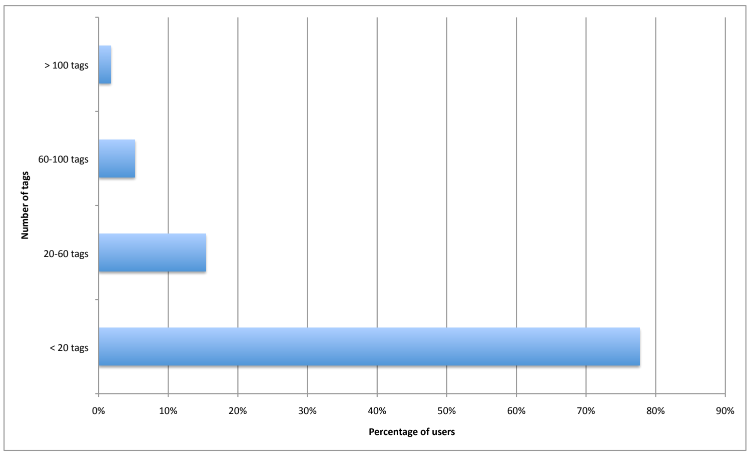

Figure 2.3: User activity on tags

remaining 14% tagged between 50 and 200 papers (high activity). Note that even the most active users only tagged a tiny portion of the whole paper set. This suggests that users have very focused and scoped interests within the much broader scientific community.

We also analysed the tagging vocabulary, i.e., how many different tags each user used to define her preferred items (Figure 2.3). We found that 77% of the users used less than 20 different tags, 15% used between 20 and 60 tags and the remaining 8% used between 60 and 120 tags. Once again, the small set of tags used by each user supports our idea that users have scoped interests.

Paper Popularity. We then studied how many users tagged the same paper (Figure 2.4)

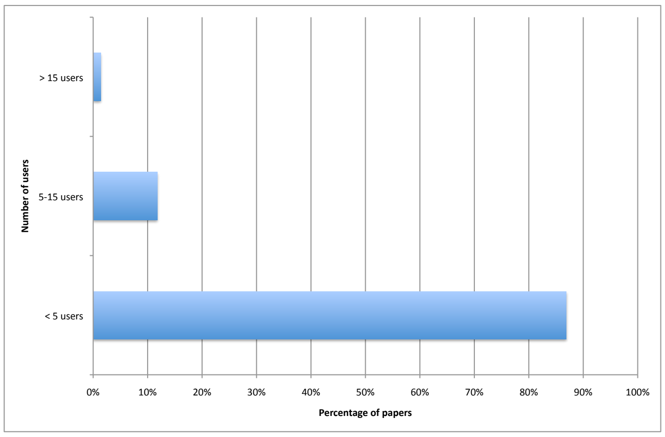

Figure 2.4: User activity on the same paper

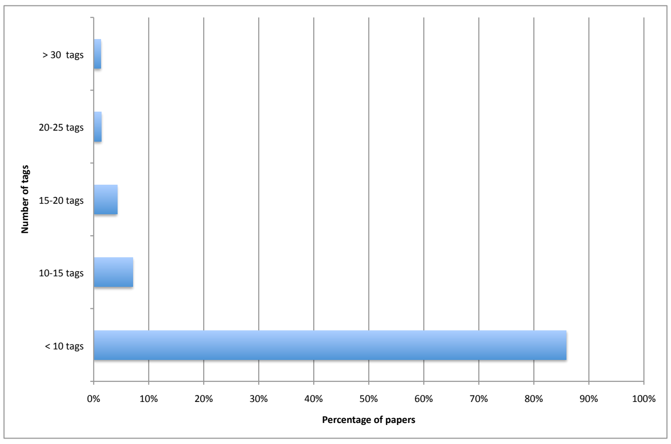

Figure 2.5: Tag activity on papers

and found that 87% of the papers were tagged by less than 5 users (low popularity), 12% were tagged by 5 to 15 users (medium popularity) and the remaining 1% were tagged by more than 15 users (high popularity). This suggests that there is a small subset of highly popular papers and a very long tail of less popular ones.

We also analysed how many different tags were used to describe each paper (Figure 2.5) and found 85% of the papers were tagged with less than 10 different tags (and more than 54% with less than 5), 14% of the papers were tagged by 10 to 30 tags and the remaining 1% were tagged with more than 30 different tags. This suggests that only a small subset of the whole folksonomy is needed to describe each of the papers, and this is true for the vast majority of them.

Tag Usage. Finally, we studied how many users used the same tags, regardless of their

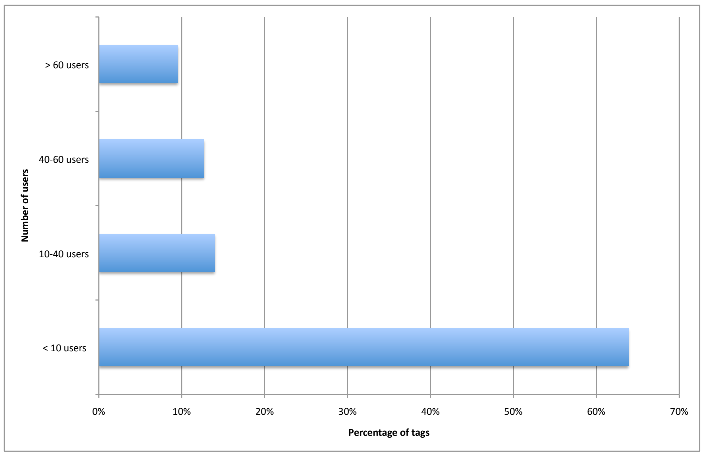

Figure 2.6: Tag usage by users

context of usage (Figure 2.6). We found that 63% of the tags were used by less than 10 users, 14% were used by 10 to 40 users, 13% were used by 40 to 60 users and the remaining 10% were used by more than 60 users. This suggests that there exists a small subset of tags that are widely used and a very long tail of less popular ones.

We also studied how spread the usage of tags was, i.e., how many different papers were associated with the same tag (Figure 2.7). We found that more than 72% of the tags were used to tag less than 20 papers, 16% were used to tag between 20 and 40 papers and the remaining 12% were used to tag more than 40 papers. This indicates that, despite the large number of tags in the CiteULike folksonomy, tags are shared by small groups of users and are used to describe a selected number of papers only.

2.2.1 Summary from Analysis

To analyse the frequency distributions of users, papers and tags reported in Section 2.2, we can refer back to an article published in an October 2004 Wired magazine by Chris Anderson [Anderson, 2006]. Here, the author defined for the first time the behaviour of the rating distribution over the Amazon and Netflix rating-based websites as a “long-tailed behaviour”. By plotting a standard demand curve (Figure 2.8[2]) showing for each item the number of users interested in it, Anderson pointed out that there existed few highly requested products and many less requested ones which gradually “tailed off” asymptotically. Moreover, the most popular 20% items represented less than 50% of all available ratings. This indicates that most of the users are interested in non-mainstream items.

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

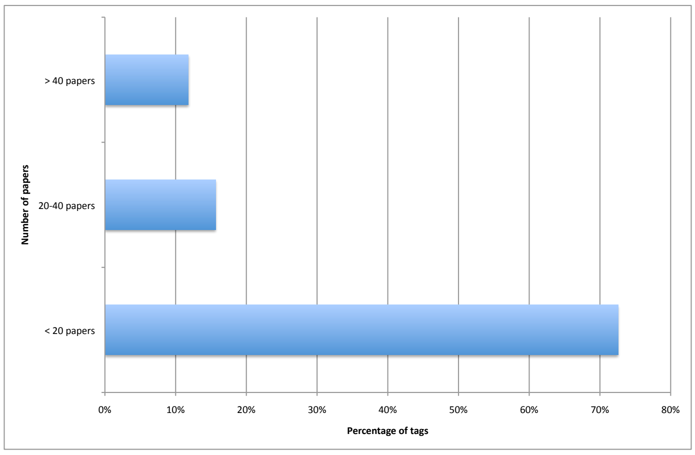

Figure 2.7: Tag usage on papers

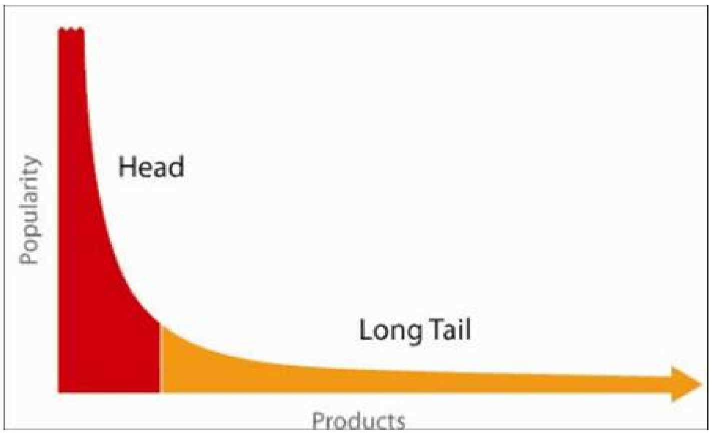

Figure 2.8: Example of a long tail distribution

Focusing on our target scenario (the CiteULike dataset), we observed two relevant long-tailed distributions.

Long tail distribution of papers' popularity: the distribution curves of papers' popularity reported in Figure 2.4 showed a small portion of popular (mainstream) papers and a long tail (roughly 87%) of (non-mainstream) papers tagged by less than 5 users.

Long tail distribution of tags' popularity: the distribution curves of tags' popularity reported in Figure 2.6 showed a small portion of highly popular tags and a long tail (roughly 60%) of tags used by less than 10 users.

The long tail distribution of papers' popularity demonstrates that most of the users are not interested in "hits" (mainstream products belonging to the head of the demand curve) but in a vast number of "niches" (unusual products belonging to the tail of the demand curve). Similarly, the long tail distribution of tags' popularity reveals a broad range of interests that goes well beyond the small set of highly popular tags. As also explained by Anderson [Anderson, 2006], this behavior is caused by the changed business model introduced by digital distribution. When consumers are offered infinite choices, the true shape of the demand curve is revealed and niche items can be as economically attractive as mainstream ones. Therefore, from a recommender system perspective, suggesting niche products is crucial to satisfy users with atypical preferences. However, the data sparsity caused by the long tail distribution leads to a significant decrease in performance of traditional CF techniques because of the *item cold start problem*. Furthermore most of the users have had little interaction with the system and thus their preferences are almost unknown. Therefore, beside the item cold start problem, recommender systems must also face a severe *user cold start problem*. A novel technique capable of suggesting even niche items to unknown users is thus called for.

2.3 Insight

To overcome the data sparsity problem described in Section 2.2.1 and to effectively suggest relevant items to users, we leverage the previous observations about the target domain reported in Section 2.2.

Exploit users' similarities over expressed interests: the vast majority of users uses only a small subset of the whole folksonomy (77% of users use less than 20 tags). This suggests that users have clearly defined interests that map to a small proportion of the whole CiteULike content and that these interests can be identified using tags. This is why we choose, as some existing works do, to calculate *users' similarities* by looking at users' tagging activity and use such similarities as part of the recommending process.

Exploit tags' similarities over tagged content: despite of the large folksonomy, each tag is used to describe a narrow subset of papers. This means users roughly agree on which tag to use on each paper. This is why we calculate *tags' similarities* by looking at which tags were associated with which papers and exploit such similarities as part of the recommending process.

Based on these two key observations, we have developed Social Ranking (SR), a technique to suggest relevant items addressing both *user* and *item cold start problems*. In the next section we describe how SR works.

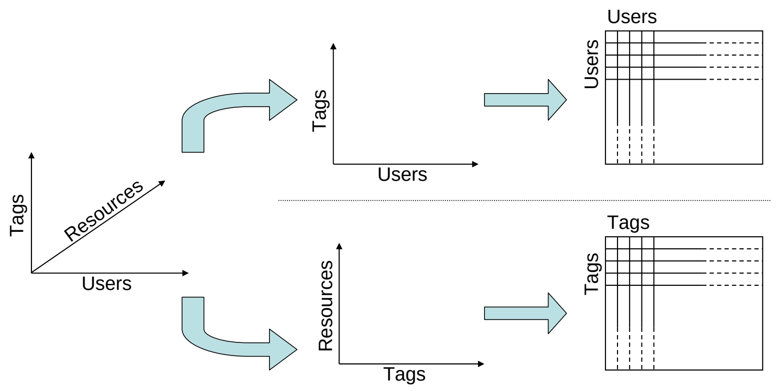

Figure 2.9: Transformation of the dataset

2.4 Social Ranking

Before going through the details of SR (Section 2.4.3), we will describe how we compute users' (Section 2.4.1) and tags' (Section 2.4.2) similarities using the information from the previous dataset analysis. We can derive a definition of similarity between users and between tags by projecting the typical tri-dimensional relationship between users, items and tags onto two two-dimensional spaces (Figure 2.9). The details of these projections for both users and tags are given next.

2.4.1 User Similarity

To measure similarities between users we analyse the set of tags they adopted to define their preferred items, using an approach similar to the one described in [Andriy et al., 2008].

We consider a simple yet effective similarity measure such that the more tags two users have used in common, the more similar they are. Note that this definition does not consider information about the tagged items. This implies flattening the tri-dimensional space of users-items-tags by projecting it onto a two-dimensional space of users-tags (Figure 2.9). This can be done because in scenarios similar to the one we have considered users interests are a rather small subset of the broader range of topics in the whole website, as demonstrated by the analysis reported in Section 2.2. As a consequence, the information we discard by performing the projection is not significant. Note that this projection can be problematic in scenarios where users have broader interests. However, such investigation goes beyond the scope of this thesis.

We describe each user u_j with a vector v_j where $v_j[m]$ counts the number of times that

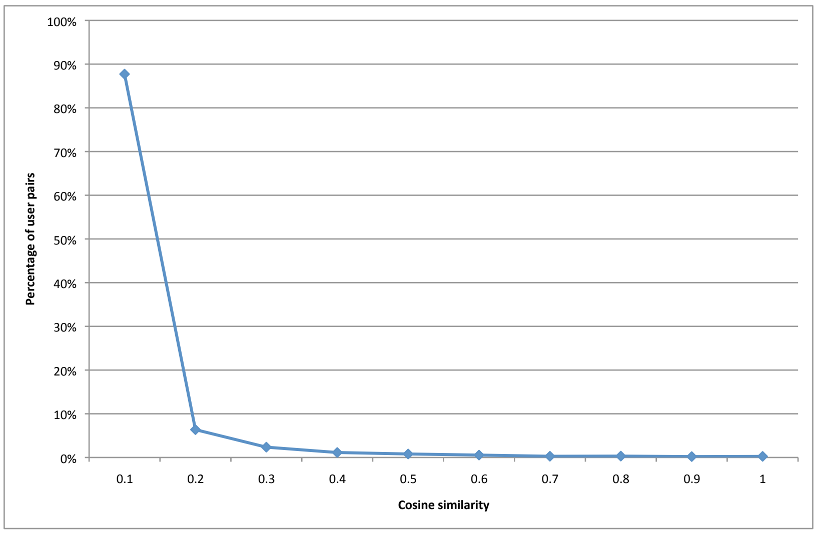

Figure 2.10: Distribution of users' similarity

user u_j used tag t_m. We then calculate the similarity $sim(u_j, u_l)$ between two users u_j and u_l as the cosine of the angle between their vectors:

$$sim(u_j, u_l) = cos(v_j, v_l) = \frac{v_j \cdot v_l}{||v_j|| * ||v_l||}$$

where $0 \leq sim(u_j, u_l) \leq 1$. We also evaluate the impact of using different values to represent the relevance of each tag for the considered user (rather than simply counting the number of usages of each tag). The results of this evaluation are reported in Section 3.4. Various similarity measures can also be used other than the cosine-based similarity [Herlocker et al., 1999]. For example, concordance-based similarity [Agresti, 1984] could be used, so that the more tags two users share, the more similar they are, regardless of how many times they have used each tag. Alternatively, Pearson Correlation and its variations, e.g., the weighted Pearson Correlation [Polat and Du, 2003] could be used. We chose cosine-based similarity because of its good performance with respect to other similarity measures, as described in [Lathia et al., 2008] and [Sarwar et al., 2001].

Figure 2.10 depicts the cumulative distribution of the similarity between pairs of users on CiteULike. The vast majority of pairs have very low similarity (below 0.1), while there exists a non-negligible amount of highly similar pairs. This suggests that on average each user can receive recommendation from a small portion of other users only. They in fact exhibit shared interests for a small set of topics (represented by tags). Our SR algorithm exploits this important characteristic of folksonomies.

2.4.2 Tag Similarity

To measure similarity between tags, we analyse the set of items they have been used on, again using an approach similar to the one described in [Andriy et al., 2008]. We consider that the more items have been tagged with the same pair of tags, the more similar the two tags are. This implies flattening the tri-dimensional space of users-items-tags by projecting it onto a two-dimensional space of items-tags (Figure 2.9). This can be done because in scenarios similar to the one we have considered tags are used by few users in the whole website, as demonstrated by the analysis reported in Section 2.2. As a consequence, the information we discard by performing the projection is not significant. As stated above, this projection can be problematic in scenarios where users have broader interests. However, we leave such investigation for future work.

We describe each tag t_j with a vector w_j where $w_j[i]$ counts the number of times that tag t_j was associated with item o_i. We then calculate the similarity $sim(t_j, t_l)$ between two tags t_j and t_l as the cosine of the angle between their vectors:

$$sim(t_j, t_l) = cos(w_j, w_l) = \frac{w_j \cdot w_l}{||w_j|| * ||w_l||}$$

where $0 \leq sim(t_j, t_l) \leq 1$. Again we also evaluate the impact of using different values to represent the relevance of each item for the considered tag (rather than simply counting the number of tagged items). The results of this evaluation are reported in Section 3.4. As above, we chose cosine-based similarity because of its good performance with respect to other similarity measures.

Figure 2.11 depicts the cumulative distribution of the similarity between pairs of tags on CiteULike. Each tag is related to only a very small subset of other tags, again suggesting that only a small portion of tags are used to describe each item. Our SR algorithm exploits this important characteristic of folksonomies.

2.4.3 Algorithm Overview

Figure 4.1 depicts the SR algorithm [Zanardi and Capra, 2008] which takes into consideration both similarity measures discussed above when suggesting interesting items to users. To understand how it works, let us consider a concrete example. A user $\overline{u}$ wants interesting items to be recommended to her (in the CiteULike case, papers). We use the term "users query" $q_{\overline{u}}$ to represent both a (proactive) search and a (reactive) recommendation. The former represents the case where the user interacting with the system explicitly defines what she is looking for, by means of user-entered tags. The latter represents the case where the system recommends items to the user, based on all tags she has used so far. In

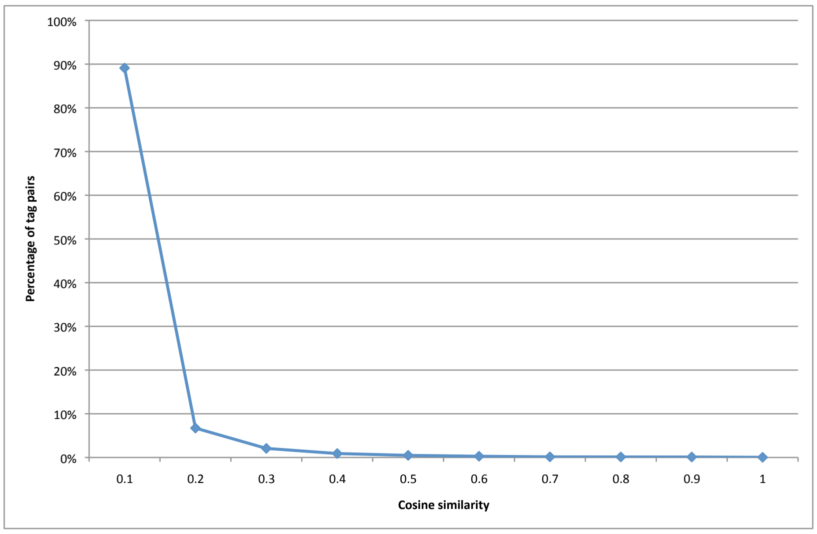

Figure 2.11: Distribution of tags' similarity

the following, we do not distinguish between the two cases, and represent a user query $q_{\overline{u}}$ as a set of query tags $q_{\overline{u}} = \{t_1, t_2, \ldots, t_n\}$.

SR uses the two similarity measures discussed above to answer to the given query. Two different steps are thus performed:

1. **Query Expansion**: the set of query tags $q_{\overline{u}}$ is expanded to include the tags $t_{n+1}, \ldots, t_{n+m}$ that are deemed similar to the query tags. The expanded tag set q^* is constructed to include, for each $t_j \in q_{\overline{u}}$, its top k most similar tags, as for the kNN strategy in traditional recommender systems. A thorough analysis of the impact of different choices of k, as well as different tag expansion methods, on the algorithm performance will be presented in Section 3.4.

2. **Ranking**: all items tagged with at least one tag from the expanded query set q^* are retrieved and suggested to the query user in a ranked list L. The ranking depends on two factors. First, the relevance of the tags associated with each item with respect to the expanded query tags. Items tagged with tags $t_j \in q_{\overline{u}}$ should be ranked higher than those tagged with $t_l \in q^* \setminus q_{\overline{u}}$. Second, the similarity of the tagging user with respect to the querying user $\overline{u}$. Items tagged by users similar to $\overline{u}$ should be ranked higher, as these users are more likely to share interests with $\overline{u}$ and thus are in a better position to recommend relevant items.

Let $U^*(o)$ be the set of users u who tagged item o with a tag from the expanded query q^*. Also, let $q^*(u, o)$ be the set of tags used by a user $u \in U^*(o)$ to tag item o and belonging to the expanded set. We define $sim(q_{\overline{u}}, q^*(u, o))$ as:

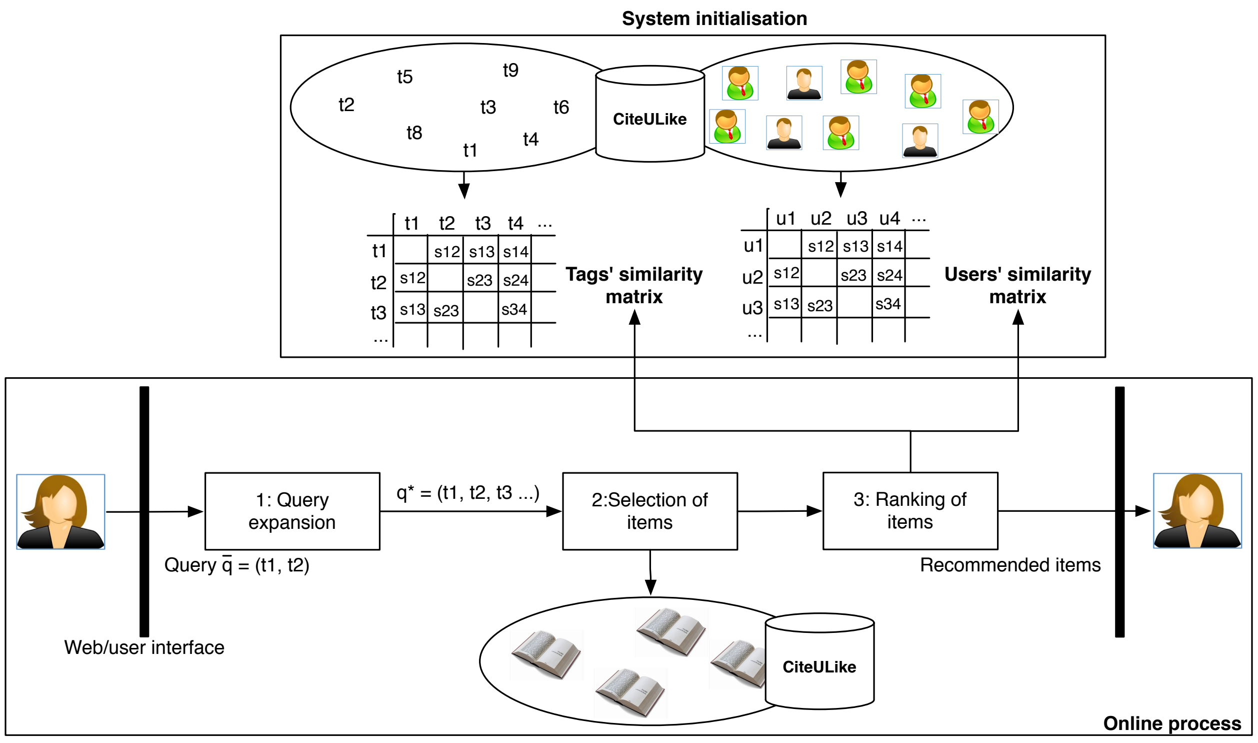

Figure 2.12: Overview of SR

$$sim(q_{\overline{u}}, q^*(u,o)) = \sum_{\substack{t_j \in q_{\overline{u}} \\ t_l \in q^*(u,o)}} \frac{sim(t_j, t_l)}{||q_{\overline{u}}||} \tag{2.6}$$

where $sim(q_{\overline{u}}, q^*(u,o))$ quantifies how relevant the expanded tags t_l associated by $u \in U^*(o)$ with o are with respect to the tags t_j belonging to the original query set $q_{\overline{u}}$. Note that $sim(t_j, t_j) = 1$ so that original tags are always considered more important in the calculation. The ranking of an item o is then computed as:

$$L(o) = \sum_{u \in U(o)} sim(q_{\overline{u}}, q^*(u,o)) * (sim(\overline{u}, u) + 1) \tag{2.7}$$

Intuitively, an item has a higher ranking if it is tagged by many users similar to the querying user (with a high $sim(\overline{u}, u)$) and with many expanded tags similar to the query tags (with a high $sim(q_{\overline{u}}, q^*(u,o))$). Note that in the formula the value of the similarity between the querying user $\overline{u}$ and each user u is increased with a "+ 1" factor. This ensures that relevant items defined by similar tags (for which $sim(q_{\overline{u}}, q^*(u,o) \neq 0)$ are still suggested to $\overline{u}$ even if they were only tagged by users outside her neighborhood (for which $sim(\overline{u}, u) = 0$).

SR outperforms traditional CF techniques. First, the tag expansion phase, which is based on all users' activity, broadens the set of recommendable items. A new item tagged with few tags have an higher probability to be recommended if one of the expanded query tags has been also used to tag it. This helps improve the algorithm's *coverage*. Recommendable items are then ranked based on both tags' and users' similarities. Therefore, even if recommenders cannot be found (e.g., for new users who are not similar to anyone), a valid ranking can still be found. Also note that the algorithm's *accuracy* is not compromised by the query expansion. Using both tags' and users' similarities guarantees that non-relevant items introduced by the query expansion will not be ranked high in the recommended list. Moreover, when users run queries with new tags (i.e., tags no one has ever used before) the algorithm runs correctly as long as the user also includes in the query other not new tags that can be used for the tag expansion.

In the following chapter, we present the results obtained when evaluating this approach.

Chapter 3

Evaluation of Social Ranking

In this chapter we thoroughly analyse the performance of Social Ranking (SR) on the CiteULike, Bibsonomy and MovieLens datasets. Before discussing the results of our experiments, we describe the metrics we used (Section 3.1), illustrate the characteristics of the datasets we experimented on (Section 3.2), and the benchmarks we used for comparison (Section 3.3). As SR relies on a number of customisable parameters, we also discuss how these were set (Section 3.4). We then analyse the obtained results (Section 3.5).

3.1 Metrics

To evaluate our recommendation technique, we adopt the standard *precision/recall* metrics computed on the recommended list, cut at the first L_{cut} best items. More precisely:

$$Precision = \frac{|relevantItems| \cap |retrievedItems|}{|retrievedItems|}$$

$$Recall = \frac{|relevantItems| \cap |retrievedItems|}{|relevantItems|}$$

Precision indicates how many relevant items are retrieved out of all returned items. In other words, it measures the accuracy of the approach. Recall indicates how many relevant items are retrieved out of all relevant items. In other words, it measures the coverage of the recommended list. The set of items that we considered *relevant* for each query will be defined in Section 3.2. Both metrics have been computed after cutting the final recommendation list at the first 10, 20, 50, 100, 500, 1000 recommended items.

In the rest of this thesis we will refer to the combined precision and recall metrics with the term *efficacy* or *performance*.

3.2 Datasets

We conducted experiments on three social tagging websites: CiteULike, Bibsonomy and MovieLens.

CiteULike is a social tagging website that aims at promoting the sharing of scientific references amongst researchers. CiteULike lets scientists label their libraries with freely chosen tags which produce a folksonomy of academic interests. CiteULike produces a daily snapshot summary of what articles have been posted by whom and with what tags up to that day. We downloaded one such archive in November 2009, containing bookmarks made between November 2004 and November 2009. We first preprocessed the dataset to remove all non-alphabetical and non-numerical tags, following the same methodology proposed by the organisers of the ECML PKDD Discovery Challenge 2009[1]. The so-pruned archive contained 41,246 users, 1,254,406 papers and 210,385 distinct tags. To further remove noise in the data, we used the p-core preprocessing strategy [Gemmell et al., 2009]. Users, items and tags are iteratively removed from the dataset to produce a smaller but denser subset where each user, item and tag occurs in at least p bookmarks. For the CiteULike dataset and in general for all the social tagging websites that we used in our experiments, the value of p has been selected so to keep a comparable amount of users, items and tags across the three datasets. This is a desirable property as experimental results will not be influenced by the datasets' different dimensions. According to the results reported in [Jschke et al., 2008], the p-core strategy does not affect the performance difference between algorithms. Note that this preprocessing only ensures that there exists a minimum amount of information for each user (i.e., each user has at least p tags/items in her profile).

The *user* and *item cold start problems* are still an issue. The final CiteULike dataset containing 2,484 users, 7,310 papers, 3,137 tags, 59,820 bookmarks and obtained by setting $p = 5$, is still composed by over 40% of *cold start users* tagging less than 10 items overall and by over 40% of *cold start items* tagged by less than 5 users. We refer to Section 3.5 for a more detailed description of the methodology which has been followed to select these parameters.

Bibsonomy is a social tagging website that aims at promoting the sharing of both scientific references and general URLs. We downloaded a snapshot of the website in June 2009, with bookmarks made between January 1989 and June 2009. We applied the same preprocessing steps described above. Also in this case we removed all non-alphabetical and non-numerical tags and then computed the p-core pruned dataset with $p = 2$. At the end we obtained a dataset with 1,360 users, 23,649 items, 11,668 tags, and 72,741

[1]http://www.kde.cs.uni-kassel.de/ws/dc09/

bookmarks.

MovieLens is a rating-based recommendation website that suggests to users movies they might like. For each registered user, the system stores information on what movies she has seen, how much she liked them (in the form of a numerical rating in the range from 1 to 5) and what tags she associated with each of them. We downloaded a snapshot of the website in January 2009, with bookmarks made between December 2005 and January 2009. Again, we removed all non-alphabetical and non-numerical tags and computed the p-core pruned dataset with $p = 2$. At the end we obtained a dataset with 1,270 users, 3,400 movies, 2,237 tags, and 23,380 bookmarks.

Table 5.1 summarizes the characteristics of the three described datasets.

Feature	**CiteULike**	**Bibsonomy**	**MovieLens**
Users	2,484	1,360	1,270
Items	7,310	23,649	3,400
Tags	3,137	11,668	2,237
Bookmarks	59,820	72,741	23,380

Table 3.1: Datasets' features

Each dataset has been used as follows during the experiments. First, we ordered the bookmarks according to their original posting date. We then performed a *temporal* split so that the first 90% bookmarks could be used for training purposes, while the most recent 10% could be used for testing. We chose a temporal split rather than a random one to reproduce the normal evolution of a real social tagging website. Table 3.2 reports, for each dataset, the size of the training (NTrain) and of the test (NTest) sets in terms of number of bookmarks. Note that the p-core pre-processing strategy has been applied before splitting each dataset into NTrain and NTest. This will cause a number of users to belong to the test set only and will stress the *cold start problem* even more. For each training set, SR has been executed to pre-compute users and tags similarities (Section 2.4). Each test bookmark has then been used as a query. The user who registered the bookmark is treated as the querying user and the tags associated to the bookmark as the query tags. This information is given in input to SR and a list of recommendations is thus produced. Precision and recall have then been measured considering as *relevant* the one item (i.e., paper, URL or movie) the test bookmark refers to. In the following we will refer to this item as the *"hidden"* item. This is similar to the approach proposed by Gemmell et al. [Gemmell et al., 2009] and Hotho et al. [Hotho et al., 2006].

Note that since every test query aims to retrieve only one relevant item in the recommendation list (whose length has been cut to values between 10 to 1000), the measured precision is always very small. What is important is not the absolute precision value, but rather the precision that SR obtains with respect to our benchmarks, described next.

Dataset	NTrain	NTest
CiteULike	53,838	5,982
Bibsonomy	65,467	7,274
MovieLens	21,042	2,338

Table 3.2: Size of training and test sets

3.3 Benchmarks

We compare the precision/recall that SR achieves with those of four benchmarks solutions:

1. Popularity-Based approach *(Pop)* - Every time a user performs a query, the system suggests her all items belonging to the dataset, sorted according to the number of times they have been tagged. The query tags are thus not taken into account, and the position of an item in the final recommendation list depends only on the number of users who tagged it (the higher the number of users who tagged it, the higher its score, the closer its ranking to the top).

2. User-based Collaborative Filtering with similarity computed with Tag usage *(CFUT)* - This technique adopts the traditional kNN Collaborative Filtering approach with users' similarity computed as follows. Each user u_j in the system is represented as a vector v_j where $v_j[m]$ counts how many times user u_j has used tag t_m (the values are normalised in the range $[0 \dots 1]$). The similarity between user u_j and user u_l is then computed as $cos(v_j, v_l)$. All items tagged by the k most similar neighbours are retrieved and ranked according to the similarity between the querying user $\overline{u}$ and the tagging users u. In our experiments we considered an unlimited value of k, thus taking all possible neighbours.

3. User-based Collaborative Filtering with similarity computed with tagged Items *(CFUI)* - This technique differs from the previous one only in the way users' similarity is computed: rather than considering tags' usage, each user u_j is represented as a vector v_j where $v_j[i]$ is 1 if u_j has tagged item o_i and 0 otherwise. The higher the number of commonly tagged items between u_i and u_j, the higher their similarity value. Also in this case we considered an unlimited value of k, thus taking all possible neighbours.

4. FolkRank *(FR)* - This technique models the system as a weighted tri-partite graph where nodes refer to users, tags and items. FR uses a random surfer strategy to recommend items to users, following the idea that an item that has been tagged with important tags and by important users becomes important itself (where "important" is intended as in the PageRank sense). Note that FolkRank was originally developed to recommend tags. We have implemented a simple modification of the algorithm so that the random walk starts from the nodes representing users and tags, differently from the original FR algorithm where the random walk starts from the nodes

representing users and items. This simple modification enables FR to recommend items instead of tags. We have included this method as a benchmark because of its accuracy [Hotho et al., 2006].

3.4 Parameter Tuning

Running SR requires the setting of a number of parameters that are dependent on the selected database. In this section, we describe a number of experiments we have conducted to decide the best setting. We have considered CiteULike dataset as target for these experiments and the selected parameters strictly depend on the characteristics of this social tagging website that have been described in Section 2.2. In this thesis, we will reuse some of the parameters which have been set for CiteULike over Bibsonomy and MovieLens datsets, given the fact that these three datasets show similar characteristics of tags' distribution and growth. A more fine grained tuning has been left as future work.

While tuning parameters, we display results of the experiments performed in terms of accuracy/coverage rather then in terms of precision/recall, as this enables us to highlight results in a more fine-grained manner. Precision and Recall display average results computed on the whole set of performed queries and they do not differ significantly across the several approaches that we considered. In other words, we decided to evaluate:

- the percentage of queries where each approach was not able to recommend the hidden item. This measure is useful to evaluate the performance of each considered technique in terms of *coverage*.
- the percentile ranking of the hidden item. This measure is useful to evaluate the performance of each considered technique in terms of *accuracy*.

The best strategy would be the one capable of minimizing the percentage of queries in which the hidden item is not suggested while maximizing instead the percentage of queries in which the hidden item is found at the top of the recommendation list.

3.4.1 Impact of Using Different Strategies to Build Users and Tags Profile

The first factor affecting SR is the strategy used to build users' and tags' profiles and to measure their similarity.

As described in Sections 2.4.1 and 2.4.2, SR relies on the information stored in the users' and tags' similarity matrices to compute recommendations. To calculate these matrices, users' and tags' profiles are created first. Different strategies can be used to build profiles, considering different methods to represent the relevance of each tag for the considered user, or the relevance of each item for the considered tag:

1. **Simple counters:** We describe each user u_j with a vector v_j where $v_j[m]$ counts the number of times that u_j used tag t_m. All tags used by u_j are considered relevant to define her interests. Similarly, we also describe each tag t_m with a vector w_m where $w_m[i]$ counts the number of times that t_m was associated with item o_i. All items tagged with t_m are considered relevant when building the tag's profile. In both situations, values have been normalized in the $[0 \ldots 1]$ range. Note that this method is the one used when we described SR in Section 2.4.
2. **Tf-idf weights:** We describe each user u_j with a vector v_j, where $v_j[m]$ represents the number of times that u_j used tag t_m divided by the total number of items tagged by u_j . Similarly, we also describe each tag t_m with a vector w_m, where $w_m[i]$ represents the number of times that t_m was associated with item o_i divided by the total number of users who used t_m.
3. **Singular Value Decomposition**. We use SVD to reduce the original dataset to a concentrated one containing only the signicant information extracted from the original data. In particular from the original $nu \times nt$ user-tag matrix and the original $nt \times no$ tag-item matrix SVD obtains an $nu \times rt$ and $nt \times ro$ reduced matrices where nu is the number of users, nt is the number of tags and no is the number of items. rt and ro are instead freely chosen parameters that indicates the reduced number of tags and items. In our experiments we considered three different values for rt and ro: 64, 14 and 5. Although increasing the values of rt and ro results in better system performance, we decided to keep them limited (64 at most) to avoid increasing system complexity.

3.4.2 Results

The experiment results suggest that even the simple counter strategy performs well when measuring users' and tags' similarities. Applying more refined techniques such as *tf-idf weights* and SVD actually lowered the performance of the system. When using SVD coverage falls (the number of unanswered query rises from 22% to 29%), while accuracy remains unchanged (the percentile ranking of the top 50% queries remains 5 for both strategies). The main reason behind the performance loss is that reducing the number of considered tags and items eliminates very important information for the recommendation process. Using SVD has in fact sense when comparing similarities between documents, where most of the words, such as connectors and articles, can be ignored. On the contrary in tagging environments every tag is meaningful. Using *tf-idf weights* on users and tags matrices does not seem to be effective either, since coverage drops of 1 percentage point and accuracy of 5 positions (the percentile ranking of the top 50% queries is 5 when we use simple counters, while it is 10 when we use *tf-idf weights*).

3.4.3 Impact of Using Different Strategies to Build the Expanded Tag Set

The second factor affecting SR is the strategy used to expand the query. We evaluated the impact of different criteria to select meaningful tags, focusing on their usage and on conceptual-semantic and lexical relations existing between words (see Section 3.4.6 for further details).

3.4.4 Activity-Based Tag Expansion

We first considered different expansion strategies based on the analysis of the usage of tags.

Average tag expansion

In this technique we consider for each tag t_j belonging to the original query $q_{\overline{u}}$, the k most similar tags according to the *cosine* similarity measure. The selected tags are then included in the expanded tag set q^*. Their weight in the final ranking calculation is then computed as the average similarity across all original query tags (Formula 2.6). As we will show in Section 3.4.5, this technique provides the best results in terms of accuracy/coverage and is therefore the one used when we described SR in Section 2.4. The main advantages of this average tag expansion are:

1. It considers equally each query tag, so that the final expanded query tag set q^* is *b*alanced. By the term *b*alanced we mean that q^* contains the k most similar tags for each tag $t_j \in q_{\overline{u}}$.
2. It gives more importance to expanded tags which are similar to more than one $t_j \in q_{\overline{u}}$.
3. Tags that are not similar to some of the query tags are penalised.

Let $q_{\overline{u}}$ be a query containing tags t_1 and t_2. Both t_1 and t_2 are expanded with their top k neighbours. If $k = 2$, q^* will be composed of 6 tags overall: the 2 query tags, 2 tags coming from the expansion of t_1 and 2 tags coming from the expansion of t_2. By keeping q^* *b*alanced, we aim at providing users with an item list L that answers the query more effectively. L will be in fact composed by an almost equal amount of items tagged with t_1

and its neighbours and with t_2 and its neighbours. Consider an example expanded tag t_3, similar to tag t_1 ($sim(t_1, t_3) = 0.8$) but not to tag t_2 ($sim(t_1, t_3) = 0.2$). By considering the average similarity value across both original tags $(0.8 + 0.2)/2 = 0.5$, we take into consideration the fact that t_3 is related only to t_1 and that it could introduce in L items which are not related at all with t_2.

Sum-score-rank tag expansion

While the *average tag expansion* takes into consideration differences between expanded and original tags and thus penalises in the final ranking of items tags that are not similar to some of the query tags, the *Sum-score-rank expansion* considers the sum of the similarity values and builds a $n$on-balanced q^*, giving priority in the final ranking of items to query tags with a bigger and highly similar neighborhood.

Let $q_{\overline{u}}$ be a query containing n tags, in this case t_1 and t_2. For each tag, we take the first $k * n$ expanded tags. If $k = 2$, we therefore take 8 expanded tags: 4 tags coming from the expansion of t_1 and 4 tags coming from the expansion of t_2. If this set contains duplicated tags, e.g. if both tags t_1 and t_2 were expanded with tag t_3, their similarity values are summed and the overall similarity value of t_3 is boosted. The whole expanded set is then ordered according to the overall similarity values and the best $(k * n)$ expanded tags are extracted. Finally, the final expanded query q^* is composed with the 2 query tags in $q_{\overline{u}}$ and the best 4 expanded tags. We make this cut to produce a set q^* as large as the expanded set produced by the *average technique*. Moreover, this choice limits the size of the expanded tag list so to focus only on the meaningful expanded tags. The final score of the tags in q^* is then rescaled in the $[0 \ldots 1]$ range. It is also important to note that the expanded query might be unbalanced. If t_1 were for example expanded only with one tag (which can happen if t_1 is not used much by users) and t_2 were expanded by 4 tags, the final tag set would contain mainly tags which come from the expansion of t_2. This could introduce in L non relevant items, highly related to tag t_2 but not to tag t_1. As shown by the experiment reported in Section 3.4.5, this results in worse performances than those obtained with the *average expansion*.

Score=1 tag expansion

This technique considers for each original tag in $q_{\overline{u}} = \{t_1, t_2\}$ the first $k \times n$ most similar expanded tags. If $k = 2$ we consider 8 expanded tags overall. Each expanded tag t_j is associated with two different scores. A *count score* that counts the number of tags in $q_{\overline{u}}$ t_j is similar to and a *sum score* that is computed as the sum of all similarity scores of t_j with the original query tags. All the expanded tags are then ranked according to the first score, while we use the second one only to sort out equally ranked tags. When calculating $L(o)$ according to Formula 2.7, the expanded query q^* is composed by the original tags and only the first $k \times n$ expanded tags. Moreover, when calculating the result of Formula 2.6 we consider that $\sum_{t_j \in q_{\overline{u}}} sim(t_j, t_l) = 1$. We do this to give to all tags the maximum

	Example expanded query tags				
Original query tags	t_3	t_4	t_5	t_6	t_7
t_1	0.3	0.6	0	0	0
t_2	0.3	0	0.8	0.8	0.6
Count score	2	1	1	1	1
Sum score	0.6	0.6	0.8	0.8	0.6

Table 3.3: Expanded query tags

Experiment	**Uncovered**
Average	5%
Sum-score-rank	10%
Score=1	8%

Table 3.4: Impact of the different tag expansion on the ranking of result in terms of coverage

similarity score.

This technique gives priority to tags which are similar to many query tags in $q_{\overline{u}}$. This avoids that the expanded tag set came from the expansion of one original tag only as it happens when using the *sum-score-rank expansion*. Furthermore, it equally weights all expanded tags, no matter what their similarity value with the original query tags is. This is done to avoid the situation that sometimes occurs with the other expansions where the tag similarity values are so low to be meaningless with respect to user similarity values.

Consider a query $q_{\overline{u}}$ performed with tags t_1 and t_2. Table 3.3 shows the similarities between the original tags and the expanded ones. In this situation *score=1* returns an expanded tag set q^* composed of the original tags t_1 and t_2, tag t_3 (similar to both), tags t_5 and t_6 (similar to t_2) and tag t_4 (similar to t_1). Consider an item o_i tagged with tags t_1, t_3 and t_5 by user u_j (i.e., $q^*(u_j, o_i) = \{t_1, t_3, t_5\}$). The value of $sim(q_{\overline{u}}, q^*(u_j, o_i)) = 3$. As shown by the experiments results reported in Section 3.4.5, the *score=1 expansion* does not achieve better performances then the *average expansion.*

3.4.5 Results

Tables 3.4 and 3.5 report the results obtained by the different implementations of SR varying the expansion strategy (and using the simple counter strategy to build users' and tags' profiles). As reported in Table 3.4, the tag expansion method that achieves the best performance in terms of coverage is the *average expansion.* Only in 5% of the performed queries it was not able to recommend the hidden item, compared to 10% of the *sum-score-rank expansion* and 8% of the *score=1 expansion.* As reported in Table 3.5, the *average*

Experiment	**Percentile**					
	5	**10**	**25**	**50**	**75**	**95**
Average	1	1	1	3	10	53
Sum-score-rank	1	1	1	9	21	77
Score=1	1	1	1	8	21	76

Table 3.5: Percentiles of the ranking of results

expansion achieves also better accuracy performance, since the percentile ranking of the hidden item is always smaller than in all other approaches, i.e., the hidden item is always returned in a higher position in the recommendation list.

3.4.6 Tag Expansion Using Dictionary-Based Approaches

A complementary approach to activity-based tag expansion is represented by dictionary-based approaches that rely on tags' similarities as statically defined by a dictionary. In particular, this approach selects the expanded tags according to their conceptual-semantic and lexical relations with the original query tags, based on the WordNet dictionary classification (http://wordnet.princeton.edu/). WordNet is a large lexical database where words are grouped into sets of synonyms, each expressing a distinct concept. These concepts are interlinked by means of conceptual-semantic and lexical relations. On the basis of the taxonomic structure of terms in WordNet, the amount of information shared between two concepts can be evaluated and a similarity value between word pairs can be computed [Seco et al., 2004]. The values obtained are comprised between 0 and 1, where 1 indicates semantical coincidence and 0 indicates no correlation. Dictionary-based techniques could be particularly useful to reduce the noise (i.e., irrelevant results) of activity-based tag expansion approaches.

We performed a wide variety of experiments trying to assess the performance of this approach. First of all, we directly compared the results obtained when using the *average expansion* (the best activity-based expansion) or a dictionary-based expansion. In the performed experiments, we also added a similarity threshold in the tag expansion phase to further reduce noise. More specifically, we decided to expand each original query tag by considering only neighbour tags with a similarity value greater than 0.1 for the average expansion and equal to 1.0 for the dictionary-based expansion (i.e., we consider only synonyms). The obtained results reveal that while the average expansion achieved better performance in terms of coverage (the percentage of queries where the hidden item could not be found was 16% for the average expansion, 21% for the dictionary-based expansion), the dictionary-based expansion was outperforming it in terms of accuracy (the percentile ranking of the top 25% queries was 10 for the dictionary-based expansion and 15% for the average expansion). This is due to the fact that only 60% of the tags belonging to the original dataset were recognized as valid words from the WordNet dictionary. In most situations the dictionary approach could thus not perform any tag expansion. When the expansion was performed instead, the approach was capable of identifying useful related tags to expand the query with. This first result suggested us to try to combine both approaches. We therefore combined the two sets of expanded tags in two different ways:

- We first performed two separate tag expansions using each one of the two presented approaches and took only the tags returned by both. However, this approach did not return good results as only 1% of the tags returned by the average expansion was also returned by the dictionary-based expansion.
- We then tried to consider all tags returned by the two expansions together. This strategy was not producing good results either, because in the vast majority of cases synonyms introduced by the dictionary-based approach only helped retrieving items not related to the query.

These results support the observations reported in [Sheung-On and Lui, 2006], where the authors studied the weaknesses of recommender systems based on simple tag matching. In particular they noted that the existence of synonyms and different inflexions of words (modifications of words due to singular/plural forms and tenses) generate linguistic inconsistencies that lower performance. Even if inconsistencies can be solved in theory by using an online dictionary (such as WordNet), in practice this is not always possible because users tend to associate different items to linguistically related words. As an example, the authors analyzed the top 5 words which are related to the "video" and "videos" tags in the Delicious dataset and found that the two sets of words were totally unrelated ("YouTube", "Conference", "Media", "Television" for "video" and "Erotica", "Public", "Sex", "Teen" for "videos"). The obtained results confirmed also the considerations reported in [Cattuto et al., 2008]. According to the study, activity-based similarity measures alone are already capable of selecting, for each considered tag, its most meaningful synonyms.

We therefore decided to use the *average expansion* alone for two main reasons:

1. It equally rewards each original tag's neighbours (it considers the top k most similar expanded tags for each original one). This property assures that, if query tags refer to different topics, an almost equal percentage of the recommended items will refer to each of them. As also underlined in [Collins-Thompson, 2009], tag expansion strategies aimed at returning balanced results lead to better recommendations.
2. It outperforms dictionary-based techniques as it considers only a meaningful subset of semantically related words which can be useful for the given query. As also explained in [Sheung-On and Lui, 2006], users tend sometimes to associate different items to linguistically related words, causing the failure of recommender systems based on simple tag matching.

Experiment	**Percentage of failed queries**				
	k=0	k=5	k=10	k=20	k=50
average expansion	36%	22%	17%	14%	8%

Table 3.6: Impact of using different values of k when performing tag expansion on coverage

Percentile	**Ranking**				
	k=0	k=5	k=10	k=20	k=50
5	1	1	5	8	10
10	1	1	5	8	10
25	1	2	5	8	10
50	3	5	15	29	40
75	10	15	30	42	54
95	53	58	72	87	96

Table 3.7: Impact of using different values of k when performing tag expansion on accuracy

3.4.7 Impact of k When Performing Tag Expansion

The final factor affecting SR is the choice of the number k of similar tags considered for each original one during the query expansion phase. Table 3.6 reports, for different values of k (from $k = 0$ meaning no expansion to $k = 50$), the percentage of queries where SR was not able to retrieve the hidden item. Table 3.7 reports instead, for each considered percentile and for different values of k, the ranking of the hidden item.

As Table 3.6 illustrates, the percentage of items not found when no expansion is performed is approximately 36%, but it quickly decreases to 17% for k equal to 10 and to 8% for k equal to 50. This result suggests that, since different users tag the same items differently, searching techniques based on user-specified query tags only (where no tag expansion is performed) are unable to find unpopular yet relevant items. On the other end, higher values of k introduce uninteresting items inside the recommendation list, as shown in Table 3.7. In fact, for higher values of k, the position in the recommended list of the hidden item indicated by the percentiles is higher (i.e., its ranking is lower). To avoid this, we decided to fix the value of k to 5. Note that an alternative approach that can be adopted during the query expansion phase consists in using a similarity threshold when selecting each tag's neighbours. In particular, instead of including in the expanded tag set the top k most similar tags for each original query tag t_j, we could include all tags t_l provided that $sim(t_j, t_l) \geq simTh$, where $simTh$ refers to a specific similarity threshold that needs to be defined a priori. Both techniques require a fine-grained tuning to decide values for k or for $simTH$. They both have possible drawbacks too: while expanded tags with a low similarity value might be introduced using a k-nearest neighbour approach, considering a similarity threshold might cause some query tags to be expanded too much

Dataset	Total n queries	UM10OM5	UM10OL5	UL10OM5	UL10OL5
Bibsonomy	1342	117	636	88	501
CiteULike	4575	1733	1423	839	580
MovieLens	2038	350	419	809	460

Table 3.8: Number of test queries performed

(if all its neighbours have a similarity value higher than *simTh*) or too little (if instead all its neighbours have a similarity value lower than *simTh*). In this thesis, we decided to use a k-nearest neighbour approach, while we leave a similarity threshold approach for future works.

3.5 Results

We now compare, for each of the three datasets under consideration, SR tuned as described in the previous sections with the benchmarks described in Section 3.3. As our approach has been devised to help *cold start users* looking for *cold start items*, we experimented with four different types of queries: 1) queries performed by active users, that is, those who have tagged at least 10 items in the training set ($UM10$); 2) queries performed by new users (cold starters), that is, those who have tagged less than 10 items in the training set ($UL10$); 3) queries where users are looking for mainstream items, that is, those tagged by more than 5 other users overall ($OM5$); 4) queries where users are looking for non mainstream items (cold start items), that is, those tagged by less than 5 other users overall ($OL5$). Note that these values have been selected only to emphasize the difference in performance between SR and the considered benchmarks depending on the characteristic of the query user and searched item. The lower these values, the greater the difference between the algorithms. We also report in Section 3.5.2 a more general analysis of the precision/recall performance of SR on the whole set of considered queries.

In both cases, we discarded from the test set all queries for which the hidden item did not belong to the training set, since none of the implemented algorithms would have been able to answer such queries successfully. Table 3.8 reports, for each dataset, the number of test queries performed in total and in each case study.

3.5.1 Precision and Recall Computed on Each Case Study

In this section we describe the results of the experiments we performed over the three different datasets Bibsonomy, CiteULike and MovieLens. In particular we report the percentage increase in precision and recall of SR with respect to the other approaches.

3.5.1.1 Bibsonomy

As Figures 3.1 and 3.2 illustrate, FolkRank is the best approach, both in terms of precision and recall, when dealing with active users searching for mainstream items. However, while the gain over standard collaborative filtering approaches (e.g., CFUT and CFUI) is considerable, the gain over SR is not significant, and it becomes lower with the growth of the recommendation list (x axis). For example, when the recommendation list is cut at the top 50 results, FR has a 19% improvement over SR both in terms of recall and precision, while the gap is 66% against CFUT.

If we now look at all the other considered query types which, according to Table 3.8, represent 90% of the test queries, the situation is completely different and SR outperforms all other approaches. As Figures 3.3 and 3.4 illustrate, FR looses 27% with respect to SR both in terms of precision and recall when dealing with active users and non-mainstream items. The gap between SR and the three approaches based on popularity (Pop) and CF (CFUI and CFUT) is consistent across all graphs. In particular in this case the gap between SR and CFUT is 85% for both precision and recall and for recommendation lists cut at the top 50 results.

The performance loss of all strategies over SR increases when considering new users and mainstream items (Figures 3.5 and 3.6). In this case the gap between SR and FR reaches 35%, while it reaches 96% between SR and CFUT (for recommendation lists of 50 elements). As Figures 3.7 and 3.8 illustrate, the query type for which SR obtains the best results is represented by queries performed by new users searching for non-mainstream items. In this case the performance gain of SR over FR and over CF strategies is almost constant and independent of the recommendation list size. If we consider the case where the recommendation list is cut at the top 50 results, SR gains a 65% and a 88% improvement over FR and CFUT respectively.

These results confirm the success of SR in answering queries from both active and new users looking for non-mainstream items (which constitute 90% of all performed queries). The performance gain in precision/recall achieved by SR confirms that our approach can deal better with both the *user* and *item cold start problems.*

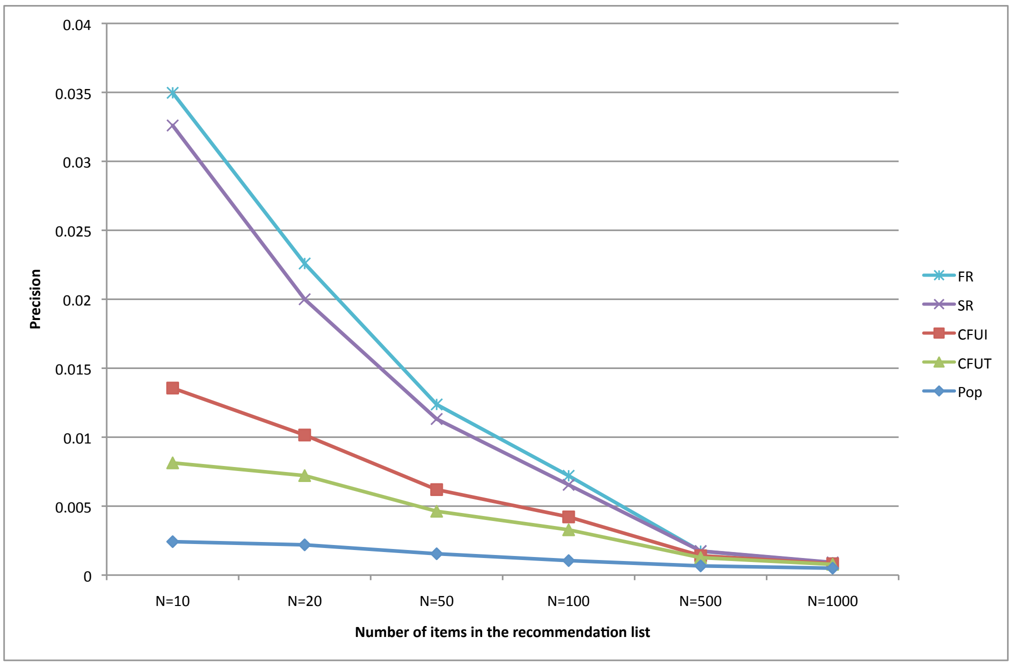

Figure 3.1: Precision for active users and popular items on Bibsonomy

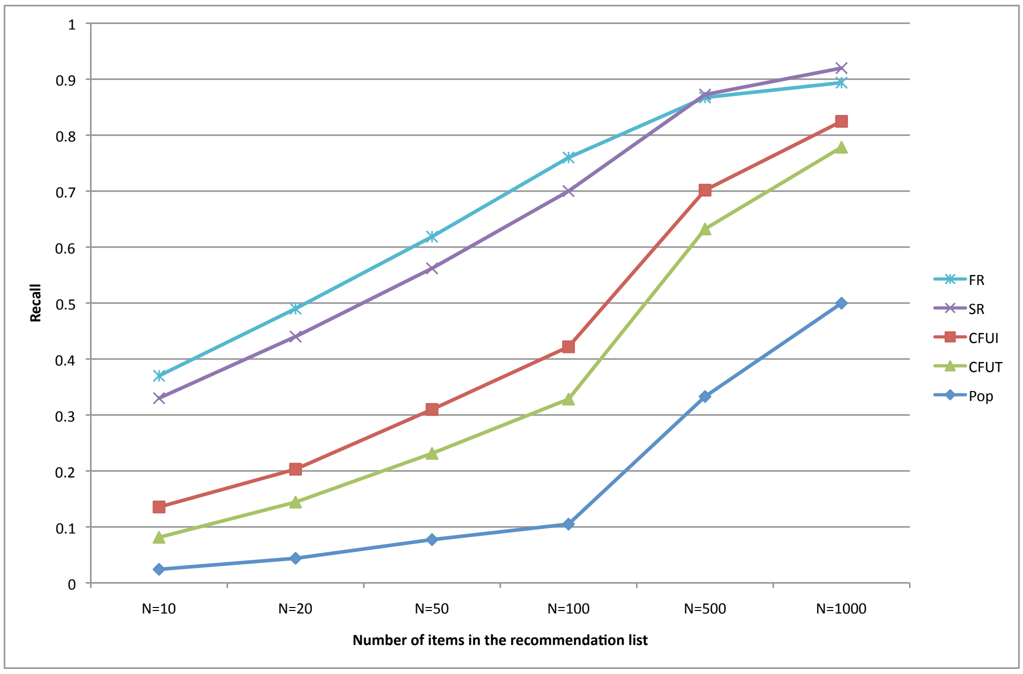

Figure 3.2: Recall for active users and popular items on Bibsonomy

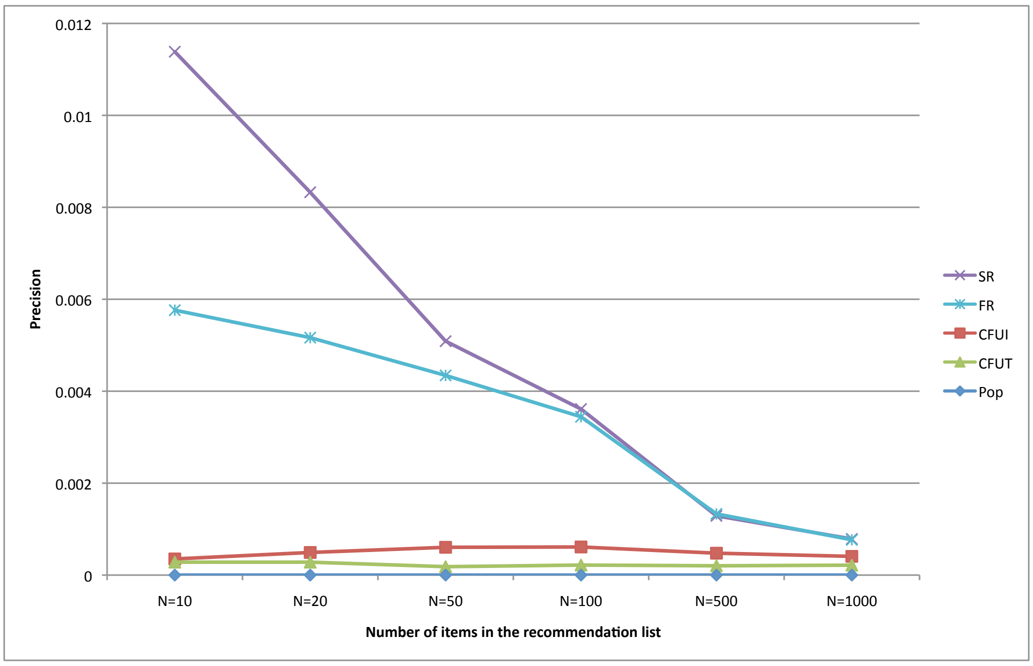

Figure 3.3: Precision for active users and unpopular items on Bibsonomy

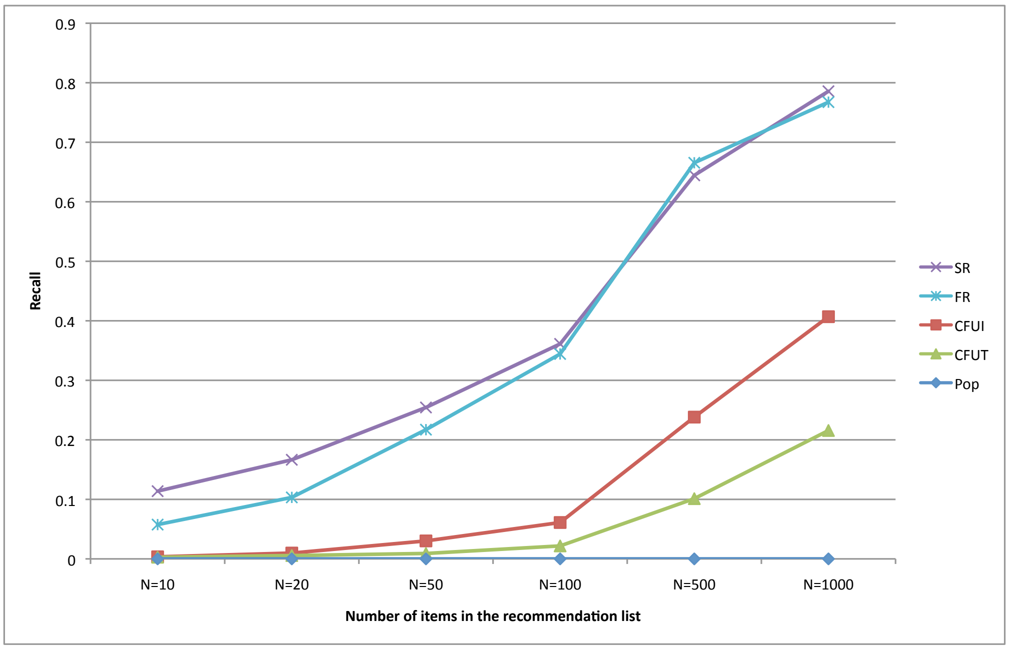

Figure 3.4: Recall for active users and unpopular items on Bibsonomy

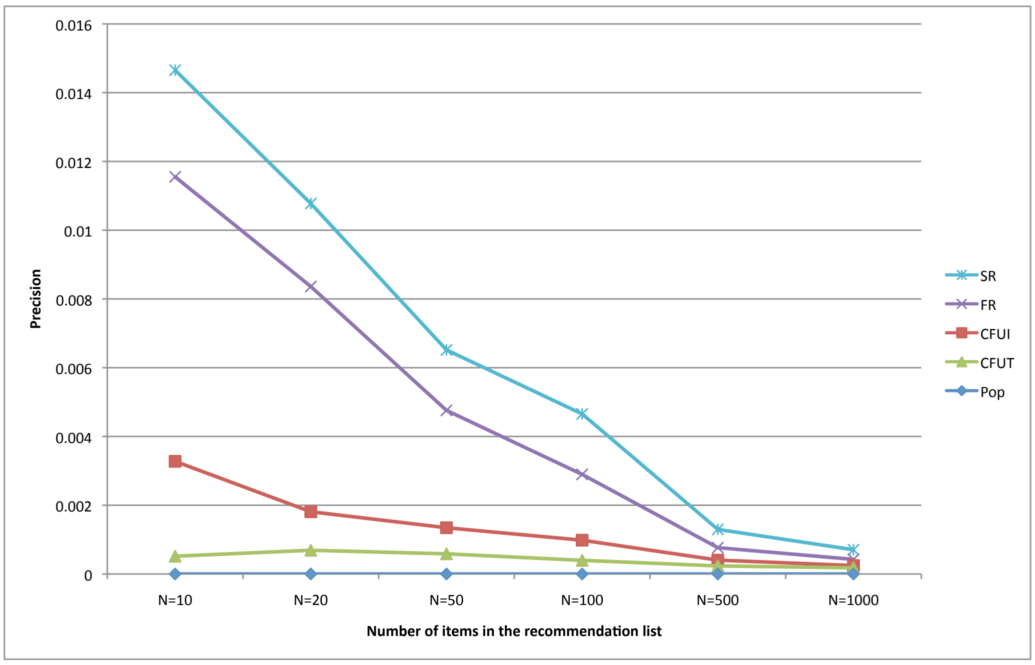

Figure 3.5: Precision for new users and popular items on Bibsonomy

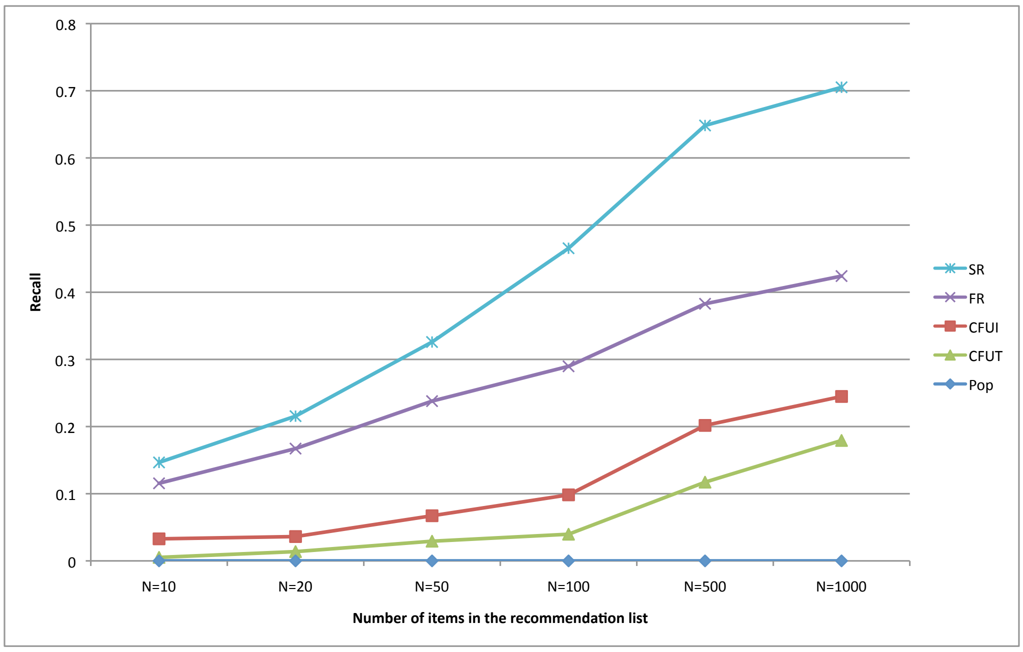

Figure 3.6: Recall for new users and popular items on Bibsonomy

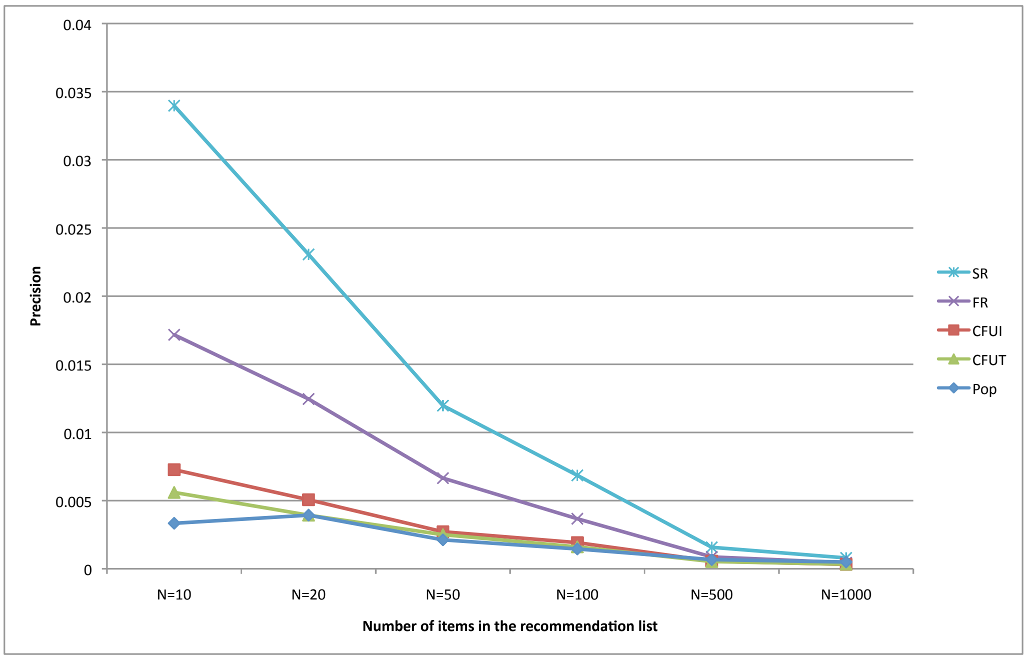

Figure 3.7: Precision for new users and unpopular items on Bibsonomy

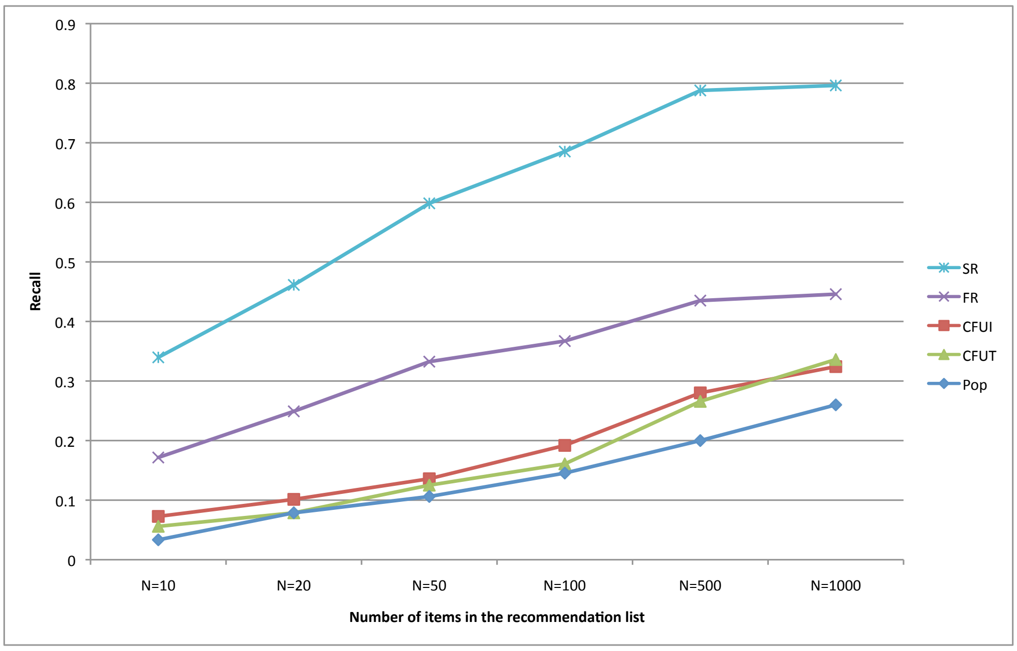

Figure 3.8: Recall for new users and unpopular items on Bibsonomy

3.5.1.2 CiteULike

We now analyse the results obtained on the CiteULike dataset. Similarly to our previous considerations, FR achieves better performance both in terms of precision and recall when dealing with active users searching for mainstream items. However, as Figures 3.9 and 3.10 illustrate, while FR obtains a substantial gain over traditional CF and Pop approaches (50% and 87% respectively in both precision and recall), the gain over SR is rather small (9% only).

We now consider all the other query types that represent 62% of the performed queries. FR is outperformed by SR when dealing with active users looking for non-mainstream items. As Figures 3.11 and 3.12 illustrate, precision and recall of SR are 15% and 96% better than those of FR and CFUT respectively (for recommendation lists of 50 elements). The gain in performance of SR increases when dealing with new users searching for mainstream items. As described by Figures 3.13 and 3.14, SR has a 27% and 91% improvement for both precision and recall over FR and CFUT respectively.

As for Bibsonomy, SR produces the best results when dealing with new users and non-mainstream items. As Figures 3.15 and 3.16 show, the performance gain of SR reaches 44% and 79% over FR and CFUT respectively for both precision and recall (for recommendation lists of 50 elements).

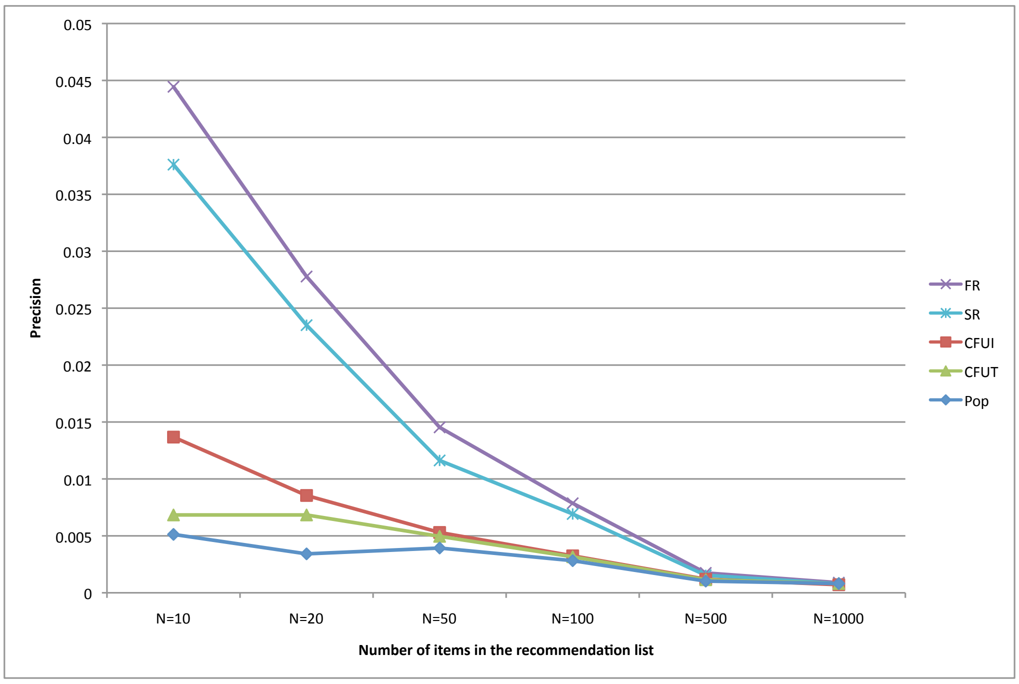

Figure 3.9: Precision for active users and popular items on CiteULike

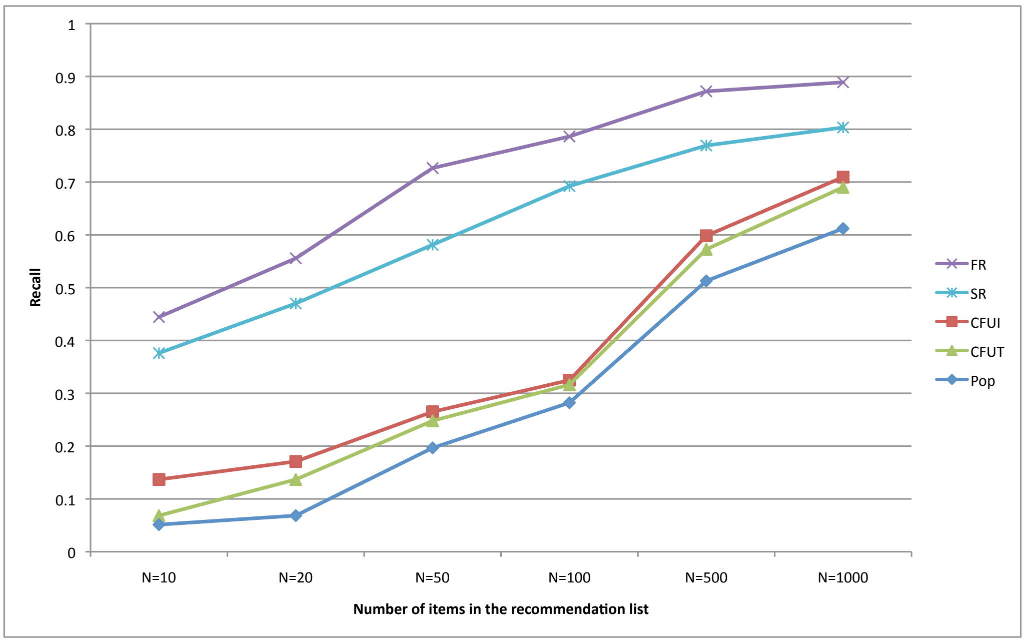

Figure 3.10: Recall for active users and popular items on CiteULike

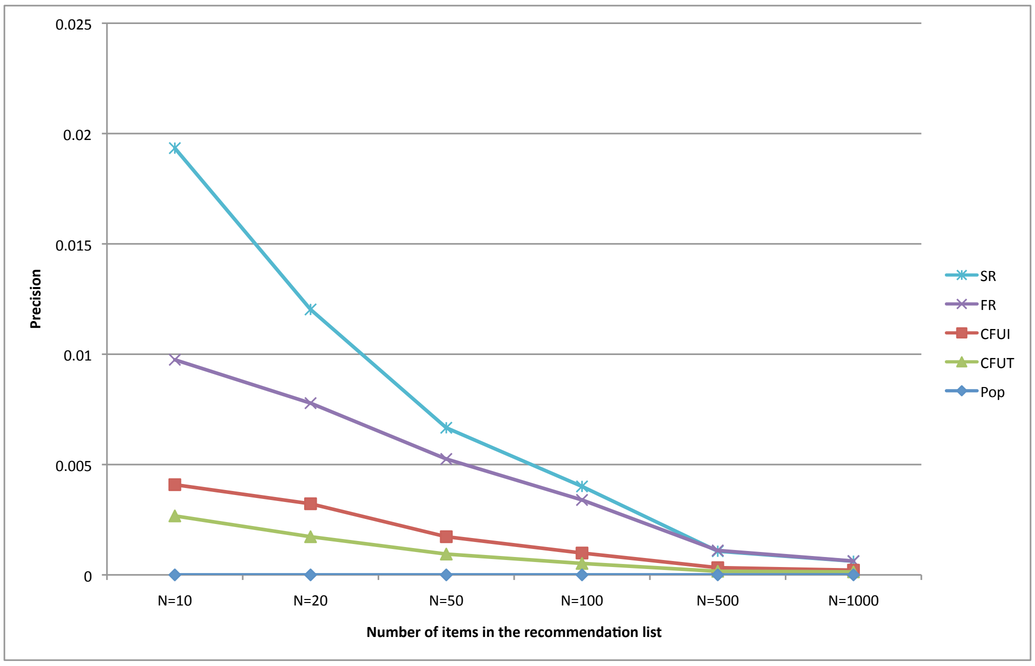

Figure 3.11: Precision for active users and unpopular items on CiteULike

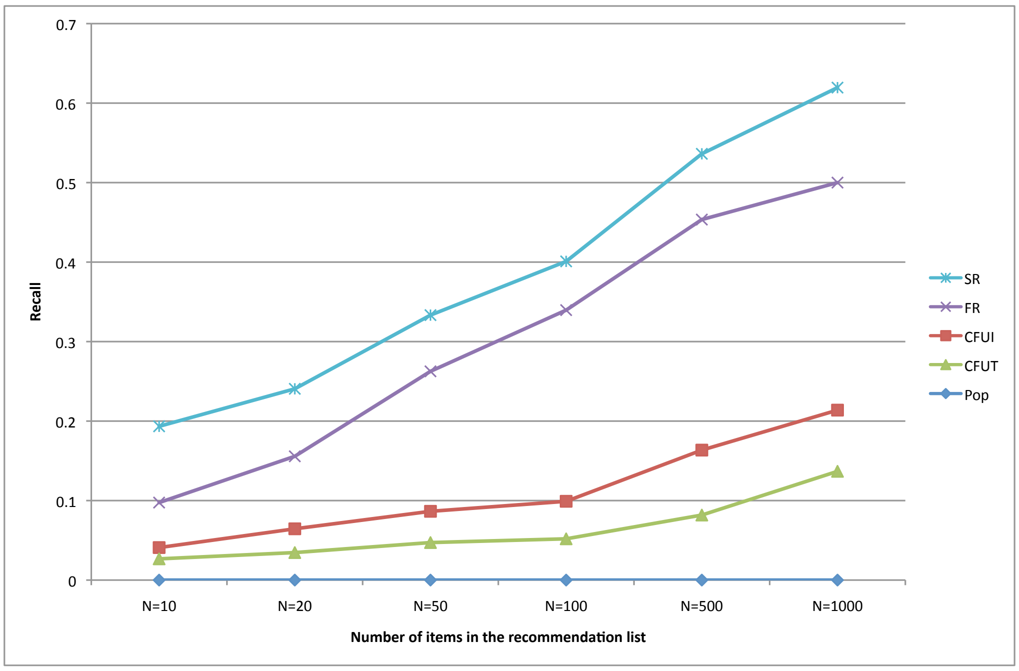

Figure 3.12: Recall for active users and unpopular items on CiteULike

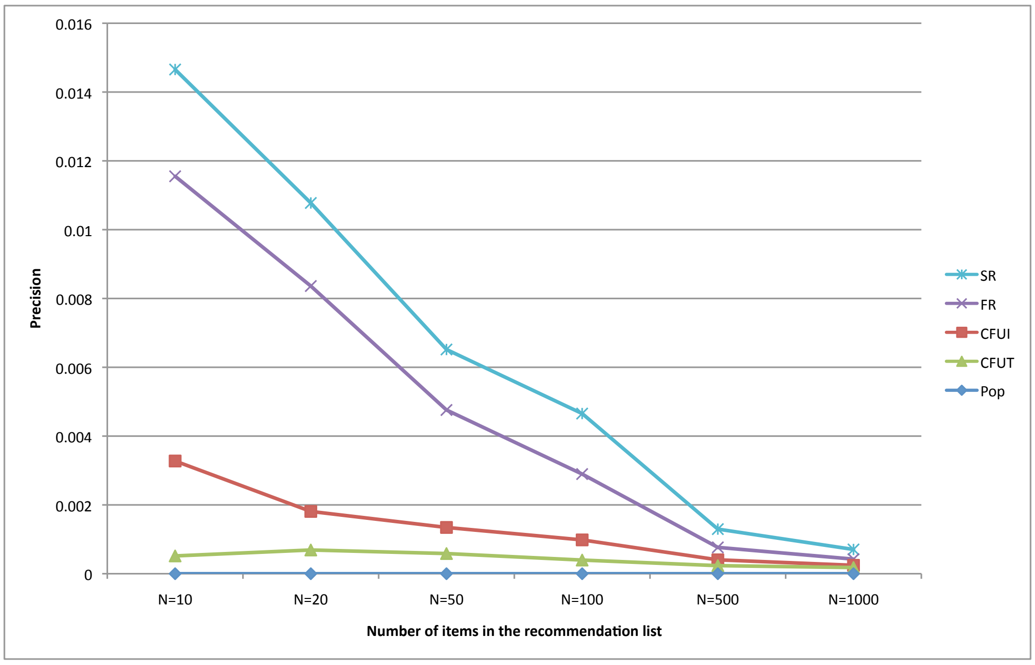

Figure 3.13: Precision for new users and popular items on CiteULike

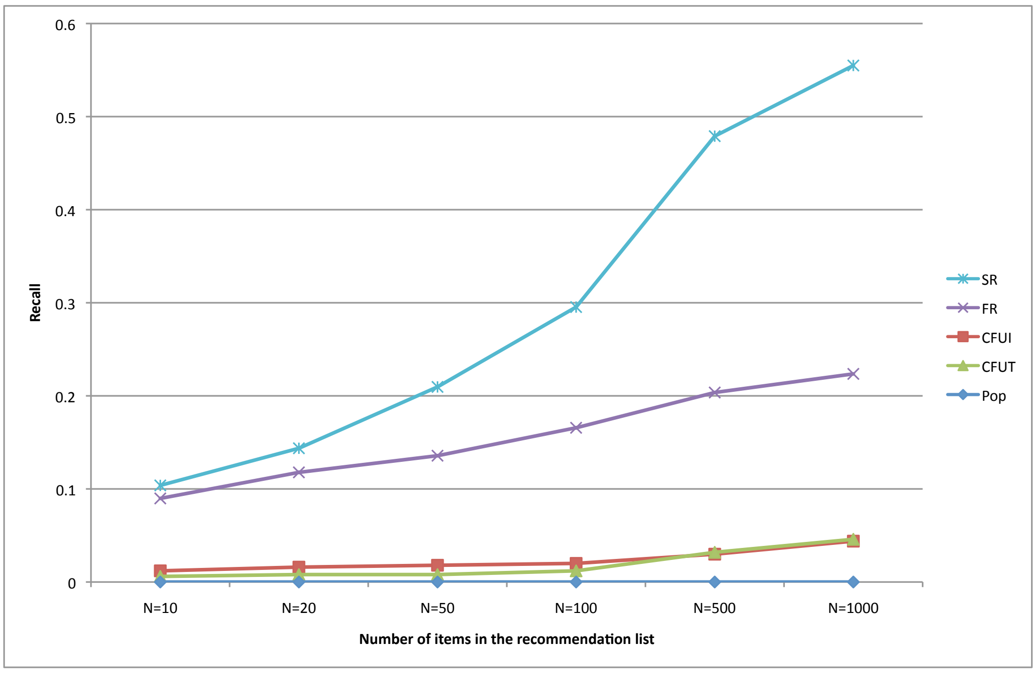

Figure 3.14: Recall for new users and popular items on CiteULike

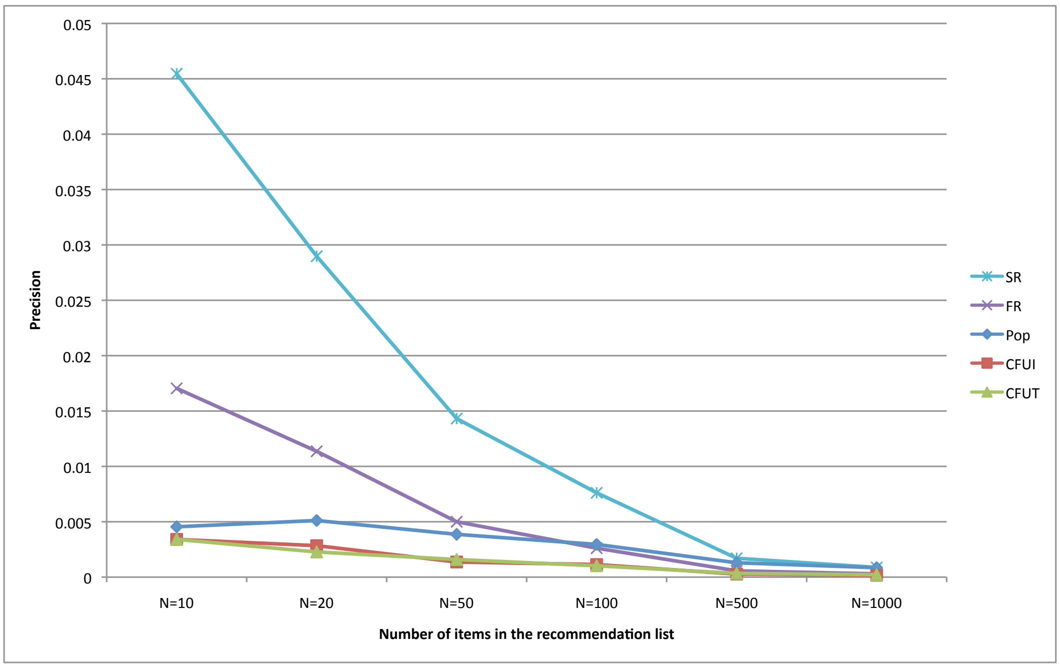

Figure 3.15: Precision for new users and unpopular items on CiteULike

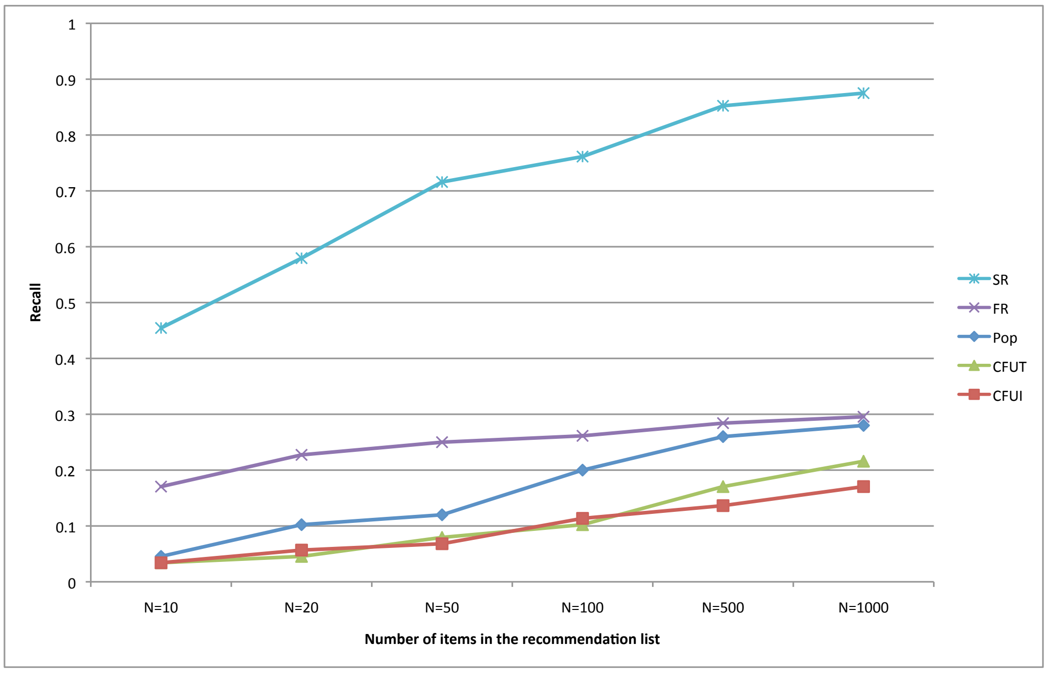

Figure 3.16: Recall for new users and unpopular items on CiteULike

3.5.1.3 MovieLens

The last dataset we analyse is MovieLens. As Figure 3.17 and Figure 3.18 illustrate, FR is again the best approach when dealing with active users looking for popular items (15% of gain over SR in both precision and recall). The same cannot be said for situations where the user is new or looking for non-mainstream items (as in 80% of the test queries). As Figures 3.19 and 3.20 illustrate, in scenarios where the user is active but the searched item is non-mainstream, SR achieves the best performance, with a gap of 14% against FR in both precision and recall. The performance gap increases even to 100% when comparing SR with CFUT.

The performance gain of SR over FR and CF increases when considering new users and non-mainsteram items. In particular, when considering new users looking for mainstream items (Figures 3.21 and 3.22), the performance improvement of SR over FR and CFUT reaches 92% and 98% respectively for both precision and recall. When considering new users looking for non-mainstream items (Figures 3.23 and 3.24), the performance improvement of SR over FR and CFUT reaches 88% and 100% respectively.

It is worth noting that on MovieLens the recall achieved by all approaches is much lower than on the other datasets we have examined. However, this behavior does not affect the performance difference between algorithms. To explain this behaviour, we have analysed the dataset in more detail and found that 70% of MovieLens' items have been tagged by less than 5 users (compared to 30% of CiteULike). This means that very little information (in terms of tags) is known about the vast majority of items. In other words, the information that recommender systems leverage to discover relevant items is not sufficient to obtain good results.

Based on the experiments conducted over the Bibsonomy, CiteULike, and MovieLens datasets, we can thus conclude that SR is the most effective technique to recommend non-mainstream items to new users, with a clear gain over other techniques. In order to have a full performance comparison, we report in the next section a more general analysis of the precision/recall performance of SR on the whole set of considered queries.

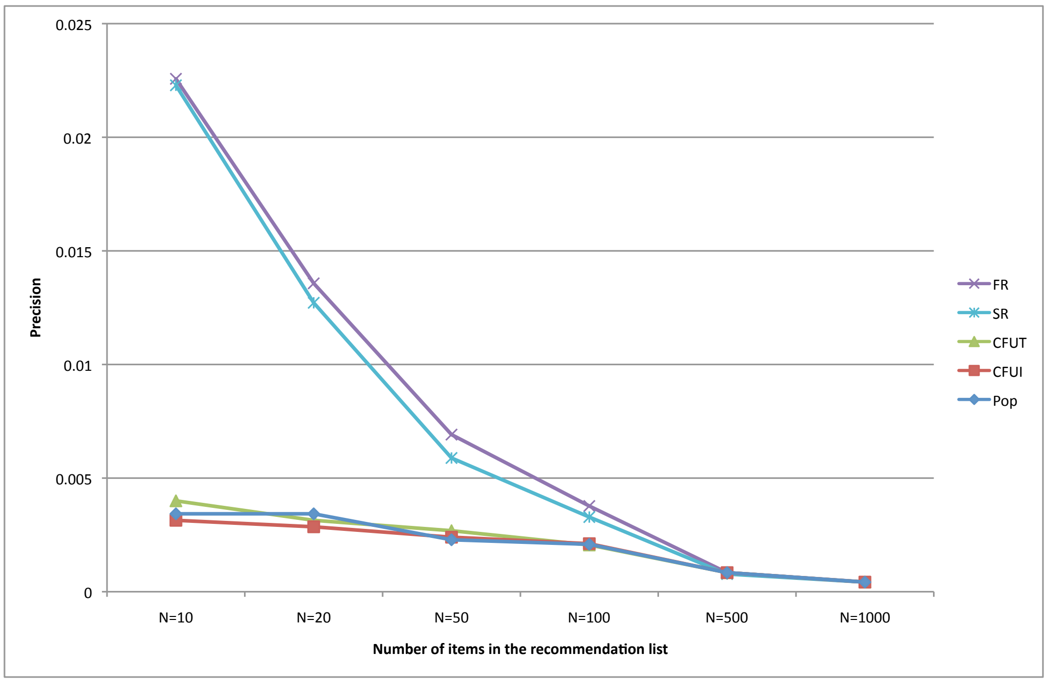

Figure 3.17: Precision for active users and popular items on MovieLens

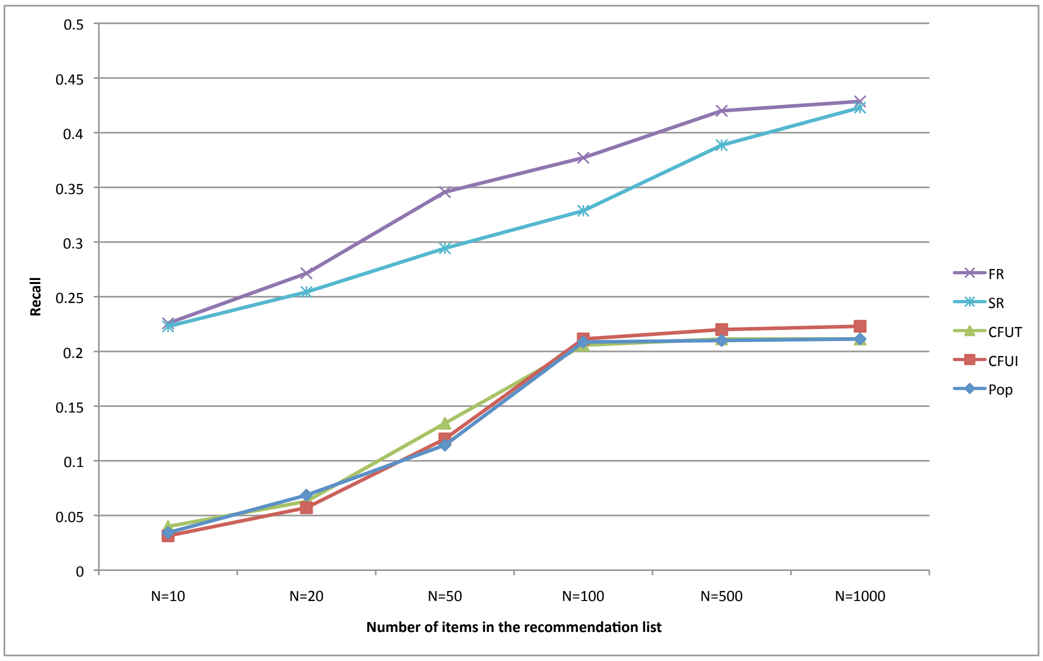

Figure 3.18: Recall for active users and popular items on MovieLens

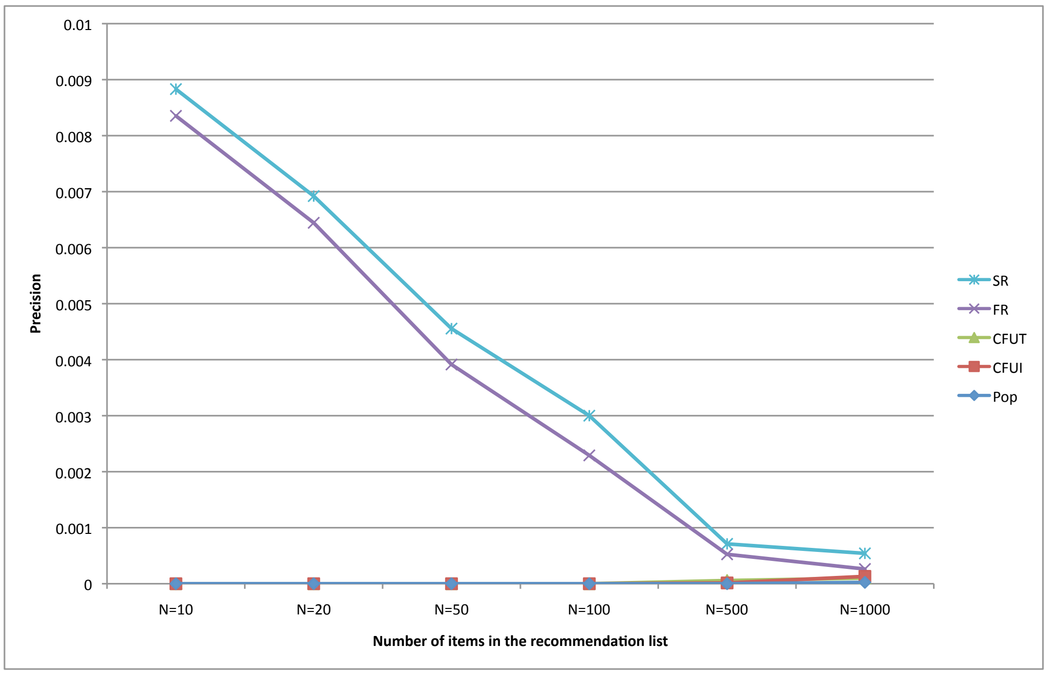

Figure 3.19: Precision for active users and unpopular items on MovieLens

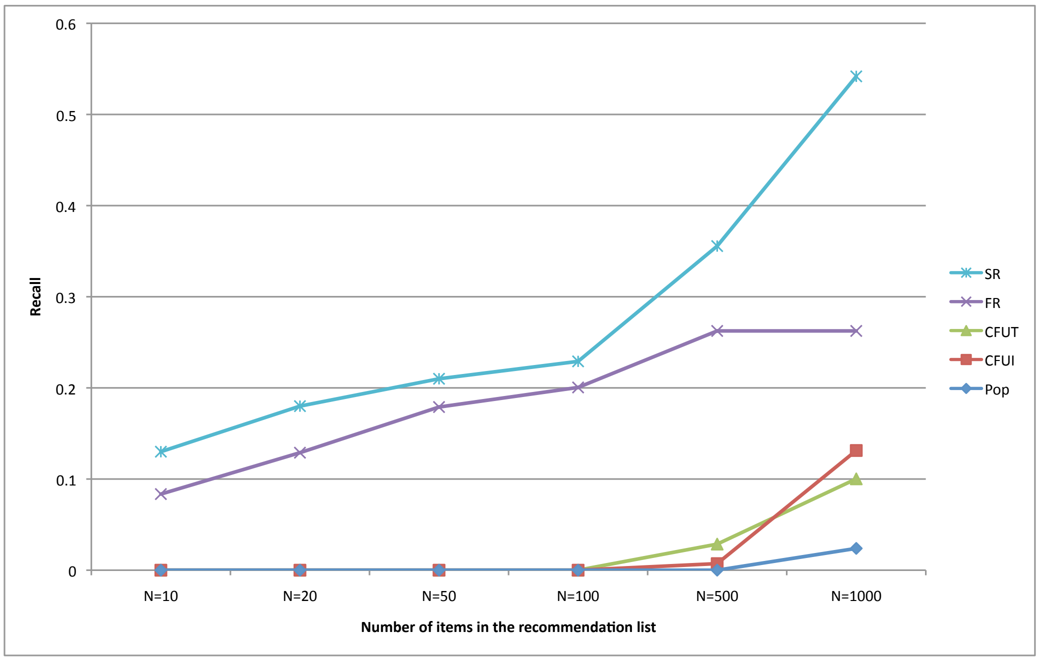

Figure 3.20: Recall for active users and unpopular items on MovieLens

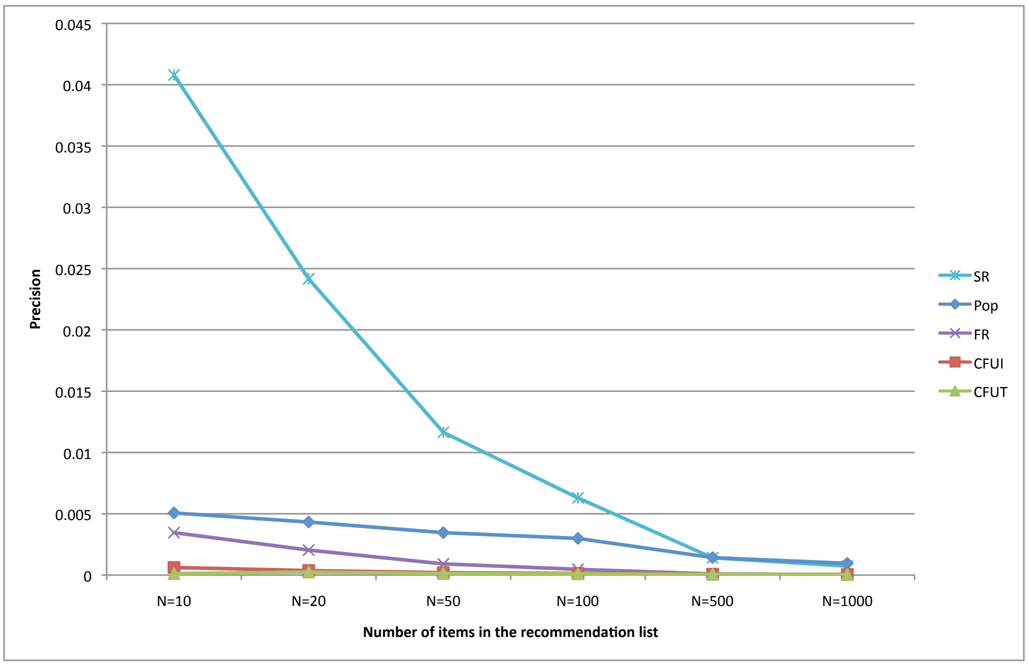

Figure 3.21: Precision for new users and popular items on MovieLens

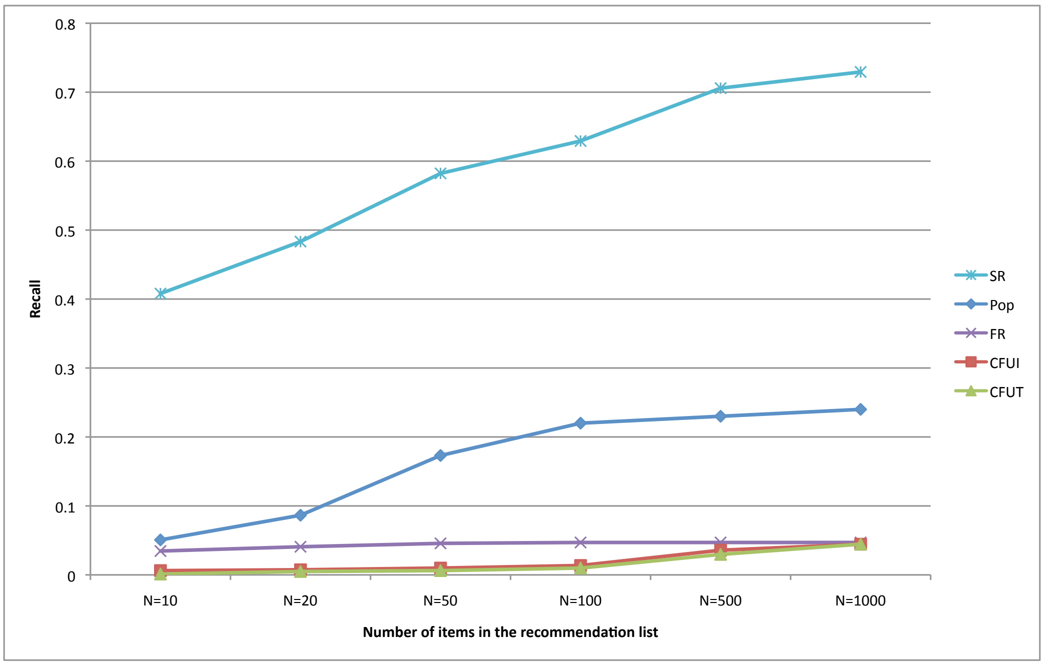

Figure 3.22: Recall for new users and popular items on MovieLens

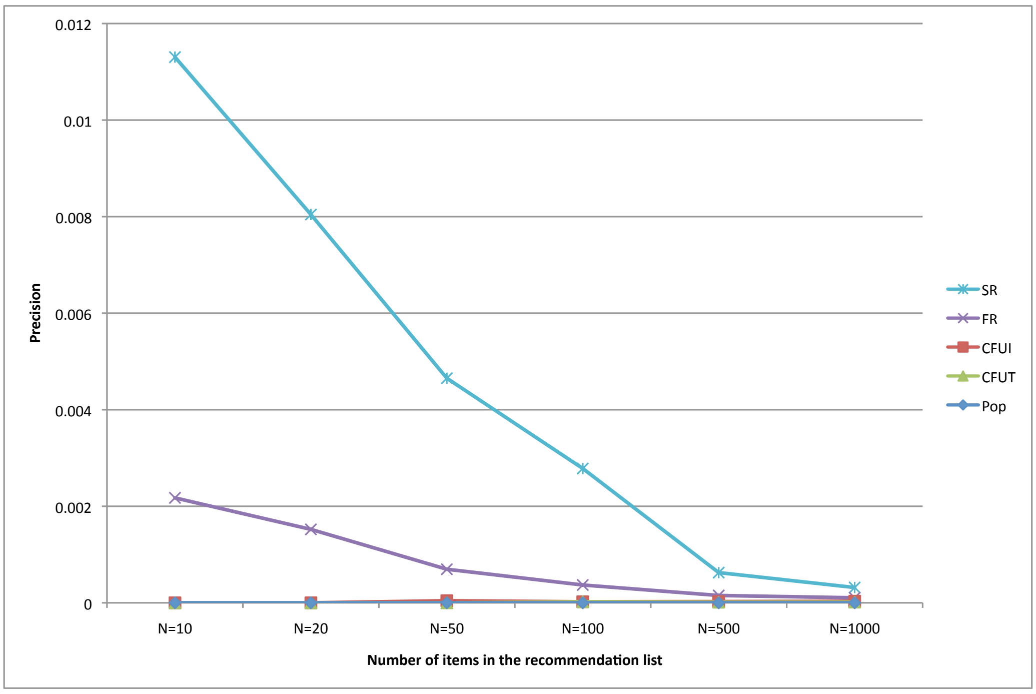

Figure 3.23: Precision for new users and unpopular items on MovieLens

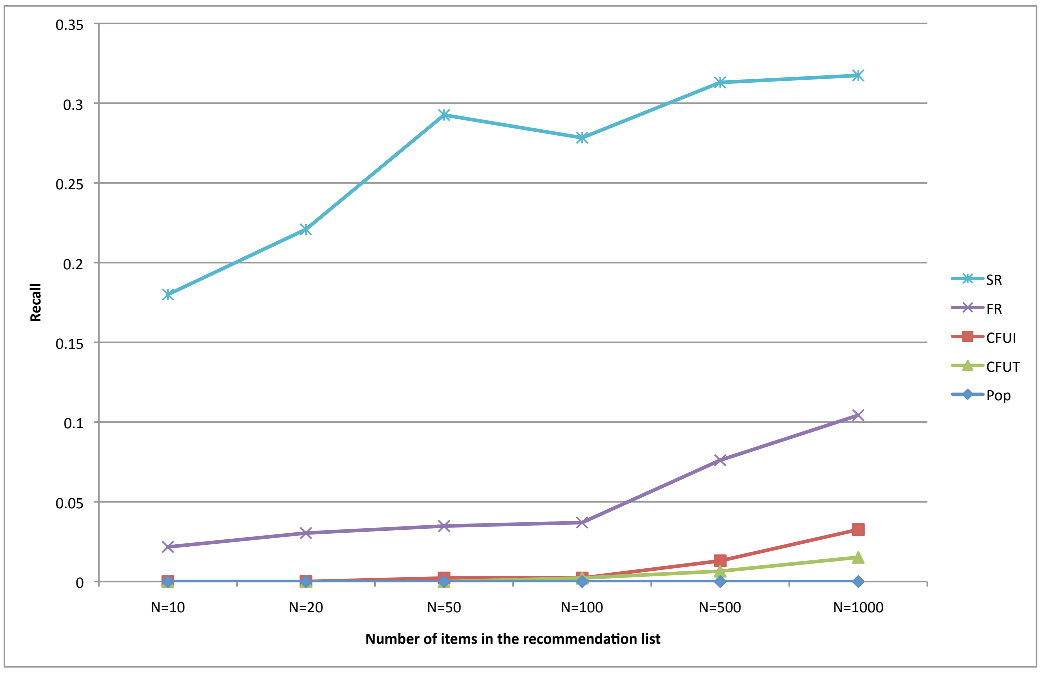

Figure 3.24: Recall for new users and unpopular items on MovieLens

3.5.2 Precision and Recall Computed on the Whole Query Set

The overall results computed on the whole set of considered queries show that SR achieves the best performance across all considered datasets, both in terms of precision and recall. For Bibsonomy (Figure 3.25 and Figure 3.26), the performance gain of SR over FR and CFUT reaches 19% and 74% respectively for both precision and recall. For CiteULike (Figure 3.27 and Figure 3.28) the performance gain of SR over FR and CFUT reaches 11% and 83% respectively, for both precision and recall. Finally, for MovieLens (Figure 3.29 and Figure 3.30), the performance gain of SR over FR and CFUT reaches 70% and 93% respectively, for both precision and recall. The reported percentages refer to recommendation lists cut at the top 50 results.

It is worth noting that the performance gain of SR over all the other considered techniques change depending on the considered dataset. This is mainly due to the different percentages of new users and non-mainstream items. In fact, the performance improvement is larger for MovieLens, which is the dataset where such values are the largest (74% and 23% of new users and non-mainstream items respectively, with respect to 47% and 4% on CiteULike and 57% and 2% on Bibsonomy).

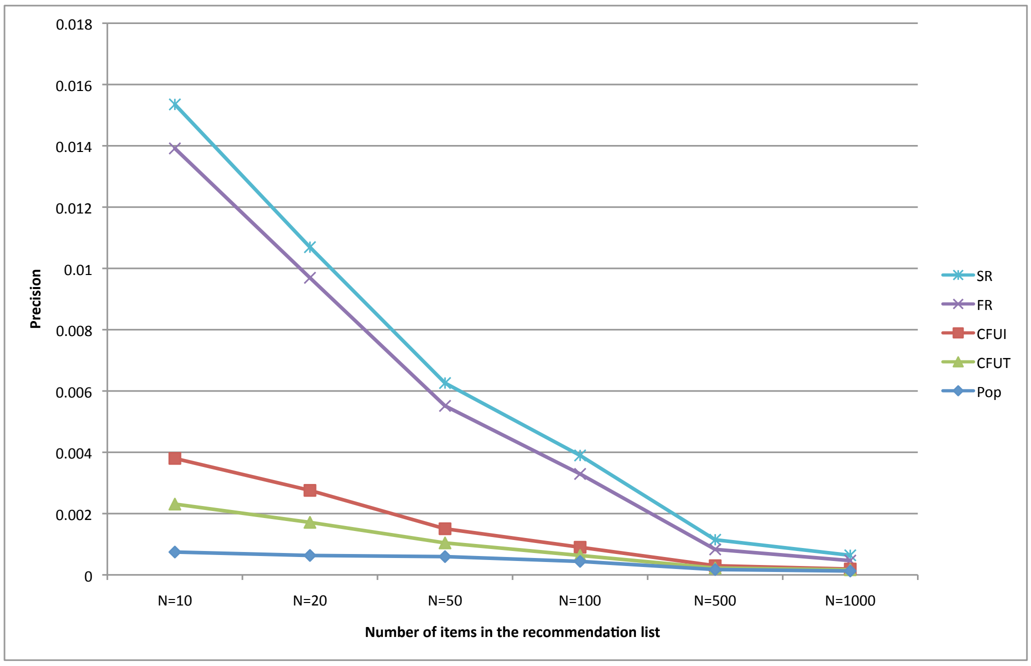

Figure 3.25: Precision on Bibsonomy

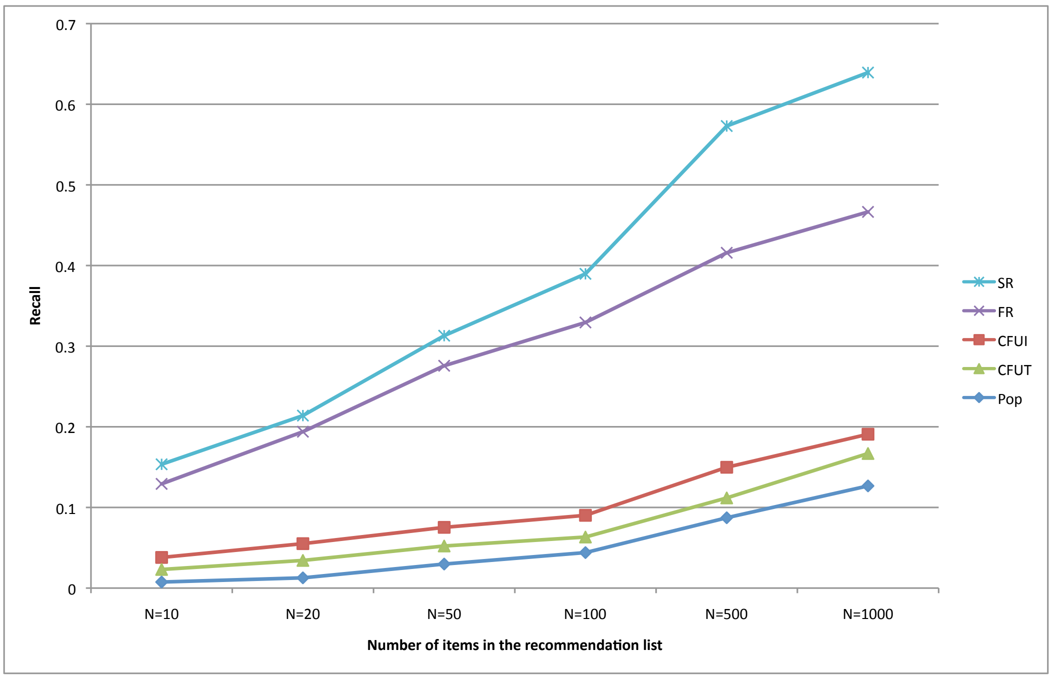

Figure 3.26: Recall on Bibsonomy

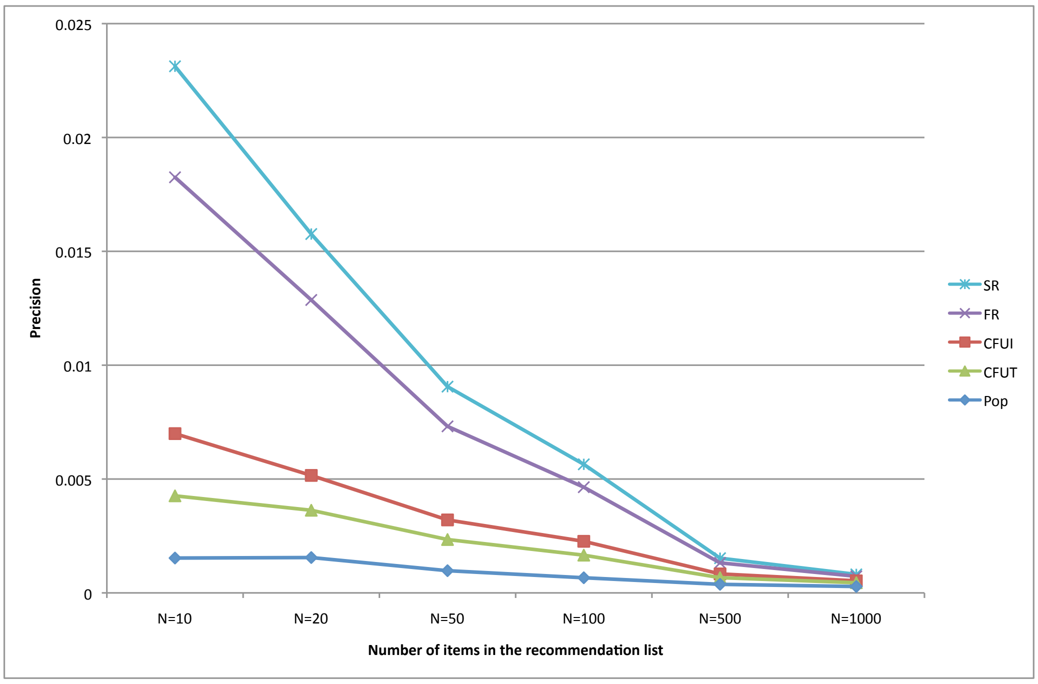

Figure 3.27: Precision on CiteULike

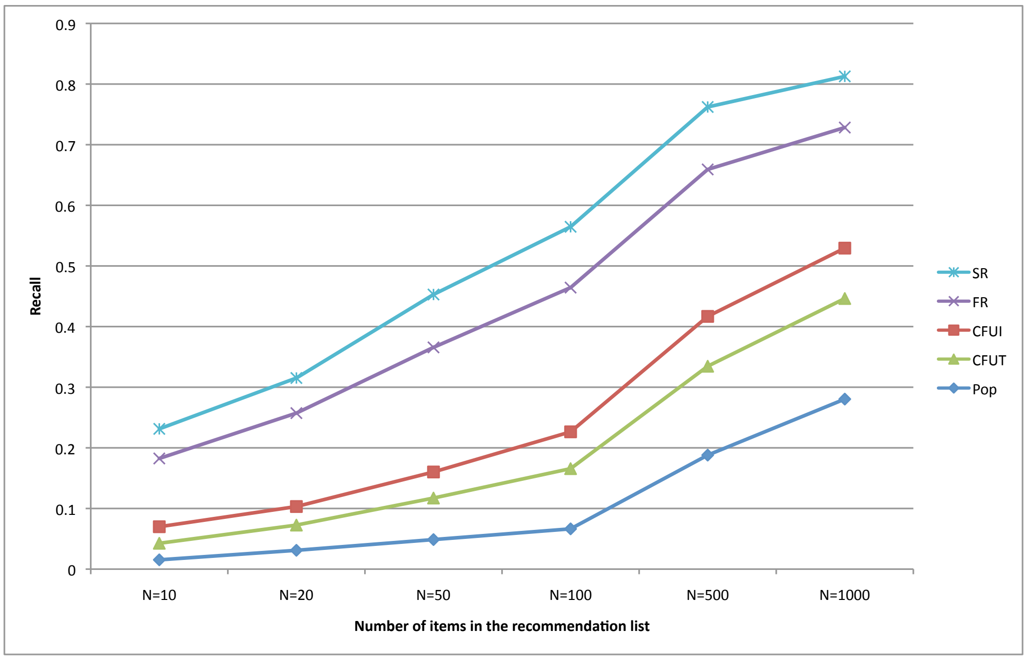

Figure 3.28: Recall on CiteULike

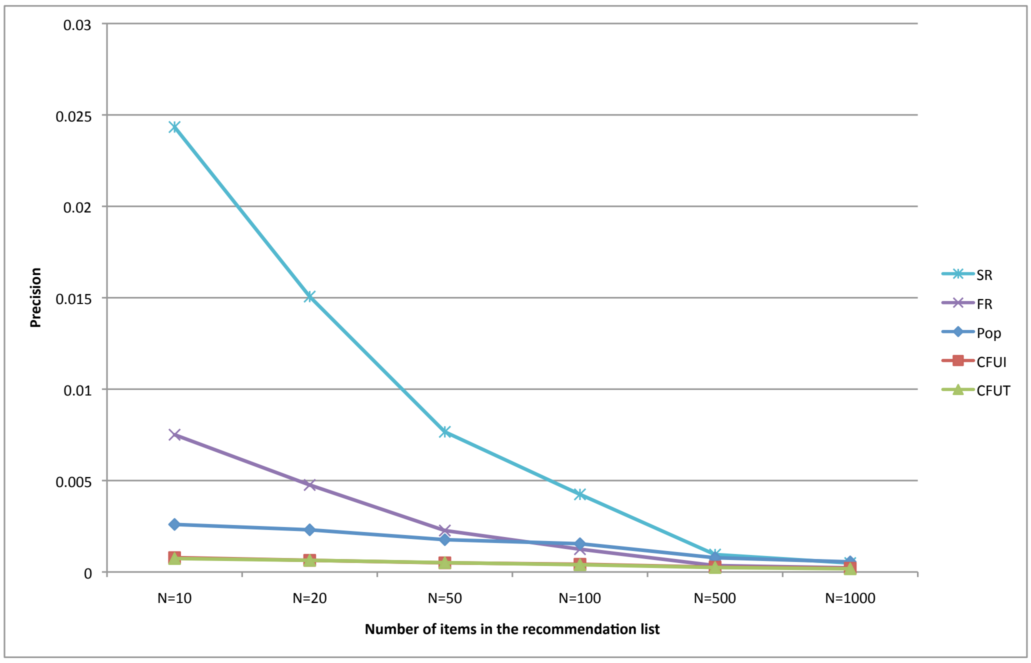

Figure 3.29: Precision on MovieLens

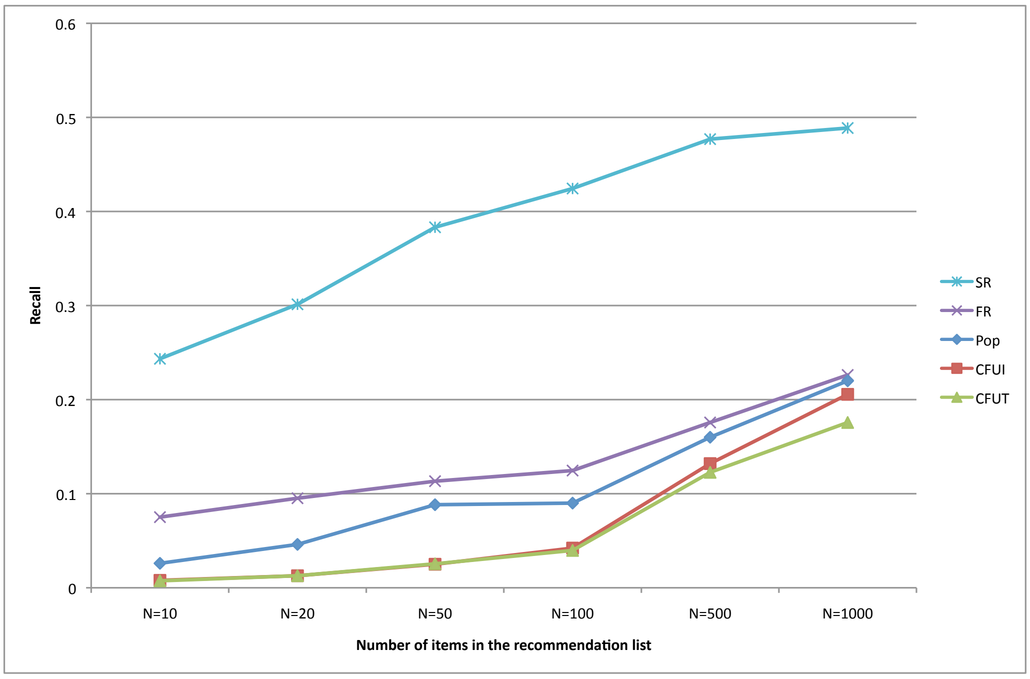

Figure 3.30: Recall on MovieLens

Chapter 4

Scalablility

Social Ranking was developed to improve the recommendation efficacy for *cold start users* and *items*. However, there is another challenge recommender systems have still to face: scalability. The amount of data recommender systems consider increases constantly due to the subscription of new users and the constant tagging activity of the community. Memory-based techniques that directly use the raw data to provide recommendations suffer particularly from this problem. However, also model-based techniques that generate offline models of the data still have to deal with it. In fact they must anyway perform periodic updates of their models to guarantee good recommendations. In the following we propose Clustered Social Ranking (CSR), a novel technique that addresses the scalability problem. CSR has a smaller computational cost and thus allows more efficient system updates then the existing methodologies. In this chapter we first introduce existing solutions to the problem of scalability for recommender systems, highlighting their limitations (Section 4.1 and 4.2). We then discuss the key principles we exploit to tackle the problem (Section 4.3) and describe the corresponding technique we developed (Section 4.4 and 4.5). We will evaluate CSR in Chapter 5.

4.1 Related Work

The problem of scalability of rating-based collaborative filtering approaches has been widely studied. A variety of different solutions have been proposed whose focus is on providing comparably accurate and complete recommendations while handling growing amounts of data. Two main directions have been followed. The first direction proposes to build a model of user ratings rather than directly using the original available information. The main advantage of these techniques is that since the model can be created offline, the online recommendation process is not affected by the data growth. This general idea has been applied to improve standard memory-based Collaborative Filtering and resulted in the creation of model-based techniques. A subset of these techniques employs matrix factorization stategies such as Principal Component Analysis (PCA) or Singular Value Decomposition (SVD) [Takacs et al., 2009, Goldberg et al., 2000, Sarwar et al., 2000] to compress the original $nu \times no$ matrix R of user-item ratings. In other words, given the matrix R, these techniques compute matrices P and Q which satisfy

$$R = P\alpha Q^T$$

where P is a $(nu \times ro)$ matrix, Q is a $(ro \times no)$ matrix and ro represent the reduced number of items considered. The matrix P can be considered as an approximation of the original matrix R in a space of ro items, with $ro \neq no$, which describes the level of interest of each user on each reduced item. The decomposed matrices are then used to compute the predicted rate for a user-item pair using the user vector extracted from P and the item vector extracted from Q. In other words, given a specific user u_j and an item o_i, the predicted rating $r'(u_j, o_i)$ is computed as:

$$r'(u_j, o_i) = \sum_{f=0}^{ro} P(u_j, f) \times Q(f, o_i)$$

Although matrix factorization techniques address the scalability issue, the computation of P and Q can be expensive. This is because the optimal value for ro considered is not fixed a-priori but decided repeating the algorithm several times and choosing the value that returned the best result. Moreover, the approximation can be challenging if some rating values are missing in the original R matrix. Unknown ratings cannot in fact be represented as zeros, as this would hinder the calculation described above. Furthermore, as also shown by the results of our experiment reported in Section 3.4, making predictions using P and Q can result in non-accurate recommendations.

Mobasher et al. [Mobasher et al., 2001] propose instead an association rule technique to recommend interesting web-pages to users. The technique is applied to transactional data, where a transaction is defined as a set of pages visited by a user in the same session. By analyzing a list of transactional logs, the authors employ the Apriori mining algorithm to capture the relationships between visited pages based on their co-occurrence patterns. In other words, the Apriori algorithm identifies groups of pages occurring frequently together in many transactions and defines a set of association rules. Given two sets of pages X and Y, each rule, expressed as $X \Rightarrow Y$, indicates that users visiting pages in X are also likely to visit pages in Y. The recommendation engine takes in input the generated collection of association rules and generates a recommended set of web pages for a user by matching the current user activity (i.e., the set of pages the user has already visited during the current session) with all association rules. Since association rules can be discovered during an offline pre-processing phase, the recommendation process scales well. However, the rule discovery phase appears to be computationally heavy and needs to be accurately tuned to avoid a common problem introduced by the mining algorithm. By focusing only on the most frequently occurring web pages in the transactional data, the algorithm fails in fact to consider rare but important web pages. Sarwar et al. [Sarwar et al., 2002] propose to apply an offline clustering algorithm over the user-item rating matrix to group users with similar profiles. The online recommendation process that they propose is inspired by traditional UBCF. Given a user u_j, the technique identifies the cluster c she belongs to and generates the predicted rating $r^{'}(u_j, o_i)$ as follows:

$$r^{'}(u_j, o_i) = \overline{r}_{u_j} + \frac{\sum_{u_l \in c}(r(u_l, o_i) - \overline{r}_{u_l}) * sim(u_j, u_l)}{\sum_{u_l \in c} |sim(u_j, u_l)|}$$

where $r(u_l, o_i)$ denotes the rating given by each user u_l belonging to the cluster c, $\overline{r}_{u_j}$ and $\overline{r}_{u_l}$ denote the average ratings of users u_j and u_l respectively and $sim(u_j, u_l)$ denotes the correlation between u_j and each user u_l. The main drawback of the described approach is that it is heavily affected by the *user cold start problem*. If the user asking for recommendations is a new user or if her profile contains few ratings, the clustering fails to classify her.

Scalability can be achieved also with a distributed implementation of traditional CF solutions. A simple decentralised implementation based on peer-to-peer networks is proposed in [Peng et al., 2004]. The originally centralised user-item matrix is first divided into fractions, called *buckets*, that are assigned to different peers. Each bucket stores, for each item-rating pair, the list of all users who rated the selected item with the considered rating. When making predictions for a user u_j, an adapted CF solution is used combining only the ratings of all users appearing in at least one bucket with u_j. A more advanced decentralized technique is proposed by MapReduce [Dean and Ghemawat, 2004]. MapReduce is a framework developed by Google to support distributed computations on large datasets. In April 2009, Apache released Mahout (http://mahout.apache.org/), a collection of scalable machine learning libraries implemented following the MapReduce paradigm. Mahout provides a distributed version of the Collaborative Filtering algorithm as well as libraries supporting different clustering algorithms (e.g., k-means, fuzzy k-means, etc.), SVD decomposition, Naive Bayes classification and frequent itemset mining.

Even if all the described solutions have been devised for rating-based environments, their general concepts could be reused in tag-based ones. However, in tag-based environments the scalability problem is worsened by the larger amount of information to process (including also tagging information). Heymann et al. [Heymann et al., 2008] showed that folksonomies are so large and dynamic that traditional recommendation techniques are no longer effective. Our goal is to define an effective and scalable recommendation algorithm for tag-based environments that provides comparably accurate recommendations even for

Approach	Offline	Online
CF	$O(\frac{nu\times(nu-1)}{2})$	$O(nu \times no)$
FR	-	$O(iterations \times na)$
SR	$O\left(\frac{nu\times(nu-1)}{2}+\frac{nt\times(nt-1)}{2}\right)$	$O(no \times nt)$

Table 4.1: Computational complexity of FR, SR and CSR

cold start users.

In the next section we analyse more in depth the scalability problem for tag-based environments by comparing the computational complexity and recommendation efficacy of state-of-the-art techniques.

4.2 Analysis of the Problem Space

We now focus on the three different recommendation algorithms considered in Section 3.3, namely CF, SR and FR. For each technique, we consider two different costs: *offline* cost and *online* cost. *Offline* cost is the cost to pre-compute all the data structures each algorithm relies on (for example, the matrix of users' similarities), while *online* cost is the cost to execute a query. Table 4.1 reports the computational complexity of each approach.

Let nu, no and nt be the total number of users, items and tags. CF requires the computation of a $nu \times nu$ matrix. The matrix stores for each pair of users u_j, u_l their similarity value that depends either on commonly tagged items or commonly used tags. The complexity of calculating the symmetric user matrix is $O(nu \times (nu-1)/2)$ (the cost of finding every pair of users). The online cost depends instead on the total number of items tagged by all neighbours of the quering user. In the worse case, the online complexity therefore is $O(nu \times no)$. As confirmed by the results of our experimental evaluation reported in Chaper 3, the low performance in terms of precision and recall makes CF unsuitable to produce effective recommendations. Moreover, its quadratic costs (offline and online) make scalability a major issue.

FR requires no offline computation instead. For each query, it traverses the tri-partite graph of users-tags-items for a number of iterations (typically 30-35) and computes a score for all items. If we indicate with na the number of edges in the graph (where in the worst case na is proportional to $nu \times no \times nt$), the online complexity of FR is $O(iterations \times na)$. Even if FR can produce effective recommendations for highly connected graphs (i.e., where users tagged many items), its online cubic cost makes it unsuitable for large folksonomies [Gemmell et al., 2009].

SR requires the offline computation of two matrices: one storing users' similarities, and another storing tags' similarities. The matrices are symmetric, thus its offline cost is $O(nu \times (nu-1)/2 + nt \times (nt-1)/2)$. The online cost depends instead on the number of items and tags used to answer queries. In the worst case where all no items have been tagged with all nt tags, the complexity is $O(no \times nt)$. Despite its good performance in producing recommendation especially for *cold start users* and *items*, SR does not scale well.

We can conclude that state-of-the-art recommendation algorithms fail in achieving both good scalability and recommendation efficacy. New solutions are thus called for. In Section 4.3, we analyze some key properties of social tagging environments that gave us insights on how to design a solution to the problem. In Sections 4.4 and 4.5 we describe the solution we devised and present the specific implementation we realized to experiment on. Note that the implementation is based on SR, even if the proposed methodology is general and other recommendation techniques could be used.

4.3 Insight

The technique we developed leverages two observations described in the following.

Leaders and Followers

According to our analysis of the CiteULike dataset, there exists a rather small proportion of users (*leaders*) who tag the vast majority of items. This suggests that it is possible to make comparably accurate recommendations by considering opinions from the set of leaders only. Studies on the Netflix and Rotten Tomatoes rating-based datasets [Amatrian et al., 2009] also confirm this idea. The authors demonstrate that it is possible to make effective recommendations without considering the whole set of existing users. They in fact apply CF techniques using only the opinions of a small set of active "experts" selected as professionals who wrote more than 250 reviews.

Domains of Interest

According to our analysis of the CiteULike dataset (Section 2.2), users are usually active only on a small portion of the whole set of existing items, thus showing focused and scoped interests within the broader scientific community. Furthermore, the analysis also shows that only a small subset of the whole folksonomy is used by users to describe each item, thus showing that users agree on which tags are useful to describe each item. These ideas are also confirmed by our studies of users' and tags' similarities described in Sections 2.4.1 and 2.4.2. Each user is similar only to a small subset of other users (who have the same scoped interests) and each tag is related only to a very small subset of other tags (that are used to tag items in the same area of interest). This suggests that the best recommenders for a target user u may just be her similar users and that we could use only this small subset to produce accurate recommendations. Let us consider a clustering of the users based on their interests. To give recommendations to a member of a cluster c we could use only the tagging information from users of the same cluster. This would also allow us to filter out noise and to improve the efficacy of the recommendation process.

4.3.1 Conclusions

Figure 4.1 depicts the approach we take to address the scalability problem in tag-based scenarios. We first identify the group of *leaders* and cluster them into domains of interest. In particular, the clusters are created according to the commonly tagged items rather than to the commonly used tags. This is done to avoid ambiguities introduced by synonyms or homonyms. Let $T(c_i)$ be the set of tags used by users in cluster c_i. Whenever a user sends a query $q_{\overline{u}}$, our solution first identifies the clusters c_i whose tags $T(c_i)$ best represent the query (further details will be given in Section 4.5). It then runs SR within the selected clusters only. Note that inside cluster c_i any technique can be used to produce recommendations. We decided to use SR for its properties we already described in Section 3. We called this implementation Clustered Social Ranking (CSR). Experiments combining our clustering recommendation approach with different techniques have been left as future work. We therefore propose to scale the recommendation process by first identifying the relevant portion of data. This does not prevent us from decentralising our approach, for example by using frameworks like MapReduce or assigning one cluster per host.

In the next section, we describe in further details our technique: we describe how leaders are identified and clustered in Section 4.4, while we illustrate how queries are associated with the best cluster to answer them in Section 4.5.

4.4 Clustering of Leaders

The literature on clustering algorithms is very rich and we could rely on different solutions to group users into categories of interests. In the following section, we describe the main characteristics of some of them, underlining how they can fit in our domain and explaining the motivation for our final choice.

4.4.1 Background Literature on Clustering

Clustering is the assignment of a set of objects into subsets (called clusters) so that objects in the same cluster are similar according to some characteristics. Each cluster is often represented by a *centroid*, i.e., an object (that may also not exist) whose characteristics

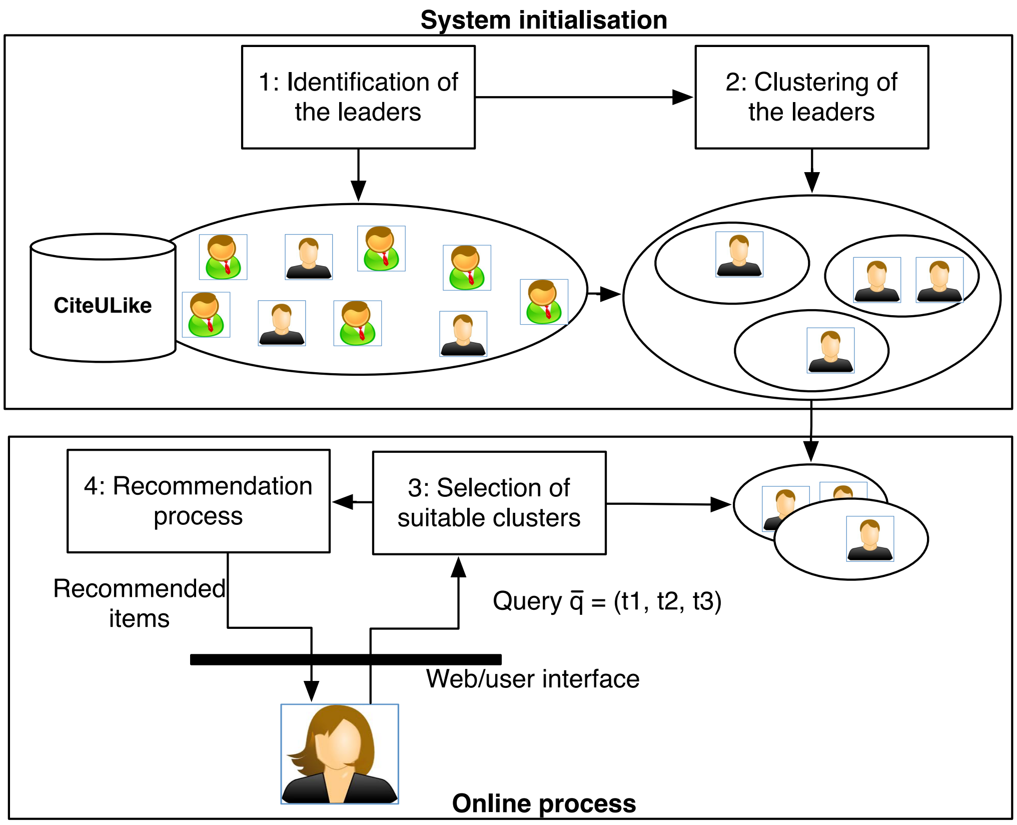
System initialisation
1: Identification of the leaders
2: Clustering of the leaders
CiteULike
4: Recommendation process
3: Selection of suitable clusters
Recommended items
Query q̄ = (t1, t2, t3)
Web/user interface
Online process

Figure 4.1: Overview of the recommendation process

are average across all objects in the cluster. Clustering algorithms [Omran et al., 2007, Schaeffer, 2007] can be divided into two major categories:

- In *vector clustering* algorithms, entities are represented as vectors that contain the score of each entity on a specific characteristic. The similarity between two entities is calculated as the distance between the respective vectors. Vector clustering algorithms can further be distinguished between *partitional clustering algorithms* and *hierarchical clustering algorithms.*
  - *Partitional clustering techniques* determine a specified number of partitions by optimizing a certain criterion function.
  - *Hierarchical clustering techniques and k-nearest neighbor pattern classification algorithms* decompose the target data into different partitions that are organised in a tree structure where leaves are the single items. The tree is constructed either top-down or bottom-up using divisive or agglomerative strategies respectively. Divisive strategies split the whole database into smaller groups itera-

tively. Agglomerative strategies group items or clusters of items iteratively.

- In *graph clustering* algorithms entities are represented as nodes in a graph with edges as mutual relationships.

We describe for each category the most widely used algorithms that have been used in the literature.

Partitional Algorithm: K-Means

One of the simplest partitional algorithm is K-Means [Manning et al., 2008]. The algorithm starts by considering a fixed set of k non-overlapping clusters, each represented by a specified centroid $c_i \in C = \{c_1, c_2, .., c_k\}$. Note that in the following we will indicate both a cluster and its centroid with the same term c_i. Each centroid is a randomly initialized vector. The algorithm then performs a series of iterations where each object x_i belonging to the original object set $X = \{x_1, x_2, .., x_n\}$ is assigned to the closest cluster, trying to maximize the within-cluster total similarity

$$W(X, C) = \sum_{c_i \in C} \sum_{x_j \in c_i} sim(x_j, c_i)$$

where sim refers to the similarity function between each object x_j and its closest centroid. At the end of each iteration, centroids are updated and moved towards the center of the group they represent:

$$c_i \in C : c_i = \frac{\sum_{x_j \in c_i} x_j}{|c_i|}$$

where $|c_i|$ is the number of objects x_j belonging to cluster c_i. The process is iterated until convergence is reached (i.e., until assignments no longer change). The K-means algorithm is very easy to implement and its linear time complexity O$(iterations \times k \times n)$ (iterations and k are usually predefined constants, [Jain et al., 1999]) makes it suitable for very large amounts of data. However, the algorithm can assign each object to one cluster only, while objects might be similar to several centroids with different levels of similarity. Fuzzy C-Means has been proposed to address this problem.

Partitional Algorithm: Fuzzy C-Means

The Fuzzy C-Means algorithm [Bezdek, 1981] is based on a fuzzy extension of the total similarity function reported in the previous section where objects can belong to more than one cluster:

$$W(X, C) = \sum_{c_i \in C} \sum_{x_j \in c_i} sim(x_j, c_i) * u_{ij}^m$$

where u_{ij}^m refers to the degree of membership of object x_j to cluster c_i. The fuzziness exponent m is a real number greater than 1 that determines the amount of overlap between groups. Note that both K-Means and Fuzzy C-Means can be performed only after specifying the number of groups k. Existing solutions to evaluate the best value of k iterate the K-Means or Fuzzy C-Means algorithms with different values of k and use different heuristics to choose the best cluster [Tibshirani et al., 2000].

Hierarchical Algorithms and K-Nearest Neighbor Pattern Classification

Hierarchical methods find clusters of similar objects by applying a sequence of agglomerative or divisive steps, with the goal of grouping (or separating) the closest pair of existing elements (or the farthest) at each iteration. The most used hierarchical algorithms are: Birch [Zhang et al., 1996], Cure [Sudipto et al., 1998], Clope [Yang et al., 2002] and Rock [Rajeev et al., 1999]. Similar algorithms have been used in template reorganization algorithms (k-neares neighbor pattern classification algorithms) adopted to improve the speed of data retrieval on database tables [Broder, 1990, Friedman et al., 1975, Faragó et al., 1993, Zhang and Srihari, 2004]. These algorithms build a dynamic clustering tree where each level represents a specific subclustering with a different granularity. They differ in the distance function which is used at each step to guide the agglomerative or divisive step and in the strategy they use to build the clustering tree. In addition, they require a great amount of memory as the clustering tree must be kept in memory and must be updated at each iteration. Moreover, since the tree does not provide a unique clustering, choosing the best level to obtain the correct partition can be challenging. Since we do not use the discovered clusters at different granularity levels (which can be useful instead when browsing information), we decided not to use these approaches in our solution.

Graph Clustering

Several algorithms have been proposed to identify clusters of users in large scale networks. These algorithms consider the objects to cluster as nodes of a graph whose edges represent the objects' relations. In particular, these techniques use methodologies from graph theory to find highly connected sets of nodes. In [Ruan and Zhang, 2008] the authors propose a modified version of the minimum k-cut algorithm to generate all the possible clusterings. Clustering are then further refined via local search techniques that merge highly connected clusters. The algorithm then finds the best clustering using the Newman and Girvan modularity function. The function decides the best clustering as the one where for each node the number of edges coming from the node's cluster is greater than the number of outgoing edges directed towards any external cluster. In [Du et al., 2007] the authors propose an approach for clustering based on the enumeration of *cliques*, where a *clique* is defined as a subset of adjacent vertices such that every two vertices in the subset are connected by an edge. In particular, the authors propose to identify all *maximal cliques*, i.e., *cliques* that are not subsets of any other *clique*. Each *maximal clique* is regarded as a clustering kernel. They then perform an agglomerative process to assign the remaining vertices to their closest kernel according to a proposed distance measure. Highly connected clusters are finally merged together.

Conclusion

The literature on clustering algorithms is very rich. We chose to experiment with the Fuzzy c-Means algorithm as it has been effectively applied in different scenarios. Moreover, it has the property that each object can be part of more that one cluster at a time. In our domain, this means that each user is allowed to belong to different clusters, as she can be interested in multiple topics. Moreover, as for K-Means, Fuzzy C-Means has small computational complexity, linear in the number of existing clusters, in the number of items clustered and in the number of iterations performed (the latter being rather small, as Table 5.3 will confirm). Experiments with other clustering techniques has been left as future work.

4.4.2 Clustering of Leaders for Clustered Social Ranking

To implement CSR in our target scenario, we first select the group of *nl leaders* as the users who tagged more than no_{high} items, following an approach similar to the one proposed by [Amatrian et al., 2009]. We experimented with different values of no_{high} as discussed in Section 5.4. Even if leaders have been selected so that they tagged most of the existing items overall, niche users who might provide useful recommendation could be ignored. We plan to consider alternative strategies to avoid this in future works.

We then modeled each *leader* u_j as a binary vector v_j over items, where $v_j[i] = 1$ if u_j has tagged item o_i. $k \approx (nl/2)^{1/2}$ clusters are initially created, with k chosen following the empirical rule of thumb described in [Mardia et al., 1979]. The initialisation of centroids is done by selecting *leaders* with non-overlapping item sets. We also experimented with a random point initialisation (i.e., each value $c_i[j]$ was set to either 0 or 1 at random). However, we found that intra-group similarity was much higher if real non overlapping *leaders*' vectors were chosen (as we will show in Section 5.4).

After this initialisation phase, Fuzzy c-Means (with $m = 2$) performs a series of iterations where each *leader* is associated to one or more clusters, depending on how well the *leader* is represented by the cluster she is being assigned to. In practice, this is computed as the cosine similarity between the *leader's* vector and each centroid's vector:

$$sim(u_j, c_i) = cos(v_j, c_i) = \frac{v_j \cdot c_i}{||v_j|| * ||c_i||}$$

Moreover, the centroid of each cluster is updated to be the mean of all *leaders*' vectors assigned to it, weighted by their similarity with the cluster. This process is repeated until the algorithm has converged, that is, the change in the similarity values between two iterations is no more than a given sensitivity threshold.

Once the clustering of *leaders* has been completed, we maintain, for each cluster c_i, the following information:

- *Item Vector*: a vector ov_i, where $ov_i[j]$ counts how many *leaders* within the cluster have tagged item o_j. In the following we use the notation $o_j \in c_i$ to indicate an object tagged by a user in cluster c_i.
- *Tag Activity Vector*: a vector ta_i, where $ta_i[j]$ counts how many times tag t_j has been used. In the following we use the notation $t_j \in c_i$ to indicate a tag used by a user in cluster c_i.
- *Tag Popularity Vector*: a vector tp_i where $tp_i[j]$ counts how many distinct users within cluster c_i have used tag t_j.

The above values have all been normalized in the $[0 \ldots 1]$ range. In the next section, we explain how these vectors are used to answer users' queries. We note that different strategies can be alternatively used to build the item and tag vectors, such as the tf-idf weighting scheme, but they have been left as future work.

4.5 Algorithm Overview

In order to answer user $\overline{u}$'s query $q_{\overline{u}} = \{t_1, t_2, \ldots, t_n\}$, CSR performs the following two steps:

1. **Query association.** First, CSR finds what clusters can best answer $q_{\overline{u}}$. To do so, it analyses the user's activity so far and the query tags. If $\overline{u}$ had little interaction with the system (i.e., she tagged less than no_{low} items, where no_{low} is not necessarily the same threshold value used to define *leaders*), the clusters-query association is based on the query tags (*tag similarity association*). If $\overline{u}$ had many interactions with the system, CSR further looks into the query tags. If $\{t_1, t_2, \ldots, t_n\}$ have been rarely used by $\overline{u}$ (that is, they have been used less than the average tag usage for $\overline{u}$), the clusters-query association is also based on the query tags. Otherwise, it is based on the items tagged by $\overline{u}$ (*item similarity association*). The underpinning idea is that, for active users who are querying the system within their well defined domain of interest, their profiles give more information about what the best cluster is (i.e., who the best recommenders are) to answer a query. However for inactive users (*cold-starters*) or users who are looking for items outside their usual domain of interest, the query tags give more information on what they are looking for.

   The clusters-query association is then performed as follows. For *tag similarity association*, we transform the query $q_{\overline{u}}$ into a vector of integers such that $q_{\overline{u}}[j]$ is equal to

1 if tag t_j is included in $q_{\overline{u}}$ and 0 otherwise. We then calculate the cosine similarity between such query vector and both the tag activity vectors ta_i ($sim_{ta} = cos(q_{\overline{u}}, ta_i)$) and the tag popularity vectors tp_i ($sim_{tp} = cos(q_{\overline{u}}, tp_i)$) for all clusters k. Groups are ranked based on the highest value between sim_{ta} and sim_{tp}, and those with the value higher than a given threshold are elected to answer the query. In all our experiments, we decided to use a threshold of zero (all clusters are elected to answer the query) to ensure the highest possible coverage, provided that accuracy will not be compromised thanks to the ranking we adopted and that is described below. A more detailed evaluation of the impact of the threshold value on both accuracy and coverage has been left as future work. In addition, note that we use both ta and tp as they provide complimentary information about the cluster. The former indicates how many different *items* are a potential match for the query. The latter indicates how many different *users* within the cluster have the same interests as the querying user (i.e., use the very same tags). Both sim_{ta} and sim_{tp} are equally important to select the best clusters.

For *item similarity association*, we compute the cosine similarity between the user's profile $\overline{v}$ and the item vector ov_i for all clusters c_i. Clusters are then ranked based on $cos(\overline{v}, ov_i)$, and those with a similarity higher than a given threshold are elected to answer the query (once again, in all our experiments, we used a threshold of zero).

For both associations, if the similarity with all clusters is zero, the query is associated with all of them. In practice, this means that we rely on all *leaders* to answer the query, regardless of their domain of interest. Note that, as *leaders* are substantially fewer than users, this is still a much lighter process than relying on the whole community as SR and traditional recommender systems approaches do. Furthermore, in all experiments reported in the next chapter, less than 3% of the queries required associations with all groups.

2. **Item discovery and ranking**. Once the clusters of best recommenders have been identified, SR is used to discover and rank items. Note that tag expansion is now performed considering only the tag similarity matrix inside the cluster. We therefore redefine q_i^* as the expanded query in cluster c_i and $q_i^*(u, o)$ as the set of tags used by a user in cluster c_i to tag item o and belonging to q_i^*. Since the domain of interest is now better scoped than when considering the whole community, we expect more suitable tags to be added to the query. To rank items, rather than considering the similarity between the querying user $\overline{u}$ and each user u_j within the selected clusters, we use the similarity computed during the association. In this way, the difference in ranking of items found within the same cluster solely depends on the query tags. If the query is associated with more than one cluster, recommendations coming from the closest cluster are ranked higher. To further mark the difference, we magnify the value of the query association by raising it to the power of a positive constant $\alpha > 1$.

The rationale for this ranking process is the following: if the querying user is a *cold starter*, or if she is known to the system but has interests in a different domain with respect to the current query, computing similarity between users would give meaningless values (in the former case) or misleading values (in the latter). In this case, only the query tags hold meaningful information for the ranking. If the user is well known to the system and she is looking for recommendations within her domain of interest, then the similarity between the querying user and the cluster should provide the same information as calculating the similarity with every *leader* in the cluster, but is cheaper to compute.

The ranking of an item o found within cluster c_i is thus computed as:

$$L(o) = \sum_{u \in c_i} sim(q_{\overline{u}}, q_i^*(u,o)) * (sim_{ASSOC} + 1)^{\alpha} \tag{4.1}$$

where sim_{ASSOC} is the item or tag association similarity and

$$sim(q_{\overline{u}}, q_i^*(u,o)) = \sum_{\substack{t_l \in q_i^*(u,o) \\ t_j \in q_{\overline{u}}}} \frac{sim(t_j, t_l)}{||q_{\overline{u}}||} \tag{4.2}$$

If an item belongs to more than one cluster, the highest ranking for the object is considered. This is done because combining the rankings from different clusters would mean considering more than once the tags associated with the item by users who belong to more than one cluster.

In the following chapter, we present the results obtained when evaluating this approach.

Chapter 5

Evaluation of Clustered Social Ranking

In this chapter we describe how we evaluated Clustered Social Ranking (CSR). We define the metrics we used (Section 5.1), illustrate the datasets we experimented on (Section 5.2), and the benchmark we used for comparison (Section 5.3). As CSR relies on a number of customisable parameters, we also discuss how these were set (Section 5.4). We finally analyse the results obtained (Section 5.5) on three different social tagging datasets. We will also compare the computational cost of CSR with the benchmarks considered (Section 5.6).

5.1 Metrics

To evaluate the efficacy of CSR, we adopted the same metrics (namely Precision and Recall) we used to evaluate the performance of Social Ranking (SR) described in Section 3.1. More precisely:

$$Precision = \frac{|relevantItems| \cap |retrievedItems|}{|retrievedItems|}$$

$$Recall = \frac{|relevantItems| \cap |retrievedItems|}{|relevantItems|}$$

Feature	**CiteULike**	**Bibsonomy**	**MovieLens**
Users	2,484	1,360	1,270
Items	7,310	23,649	3,400
Tags	3,137	11,668	2,237
Bookmarks	59,820	72,741	23,380

Table 5.1: Datasets' characteristics

5.2 Datasets

We conducted experiments using the same social tagging websites also used to evaluate the performance of SR: CiteULike, Bibsonomy, and MovieLens. A thorough description of these datasets can be found in Section 3.2. Table 5.1 summarizes their characteristics.

5.3 Benchmark

We compared the precision/recall values that CSR achieves with those of the five benchmarks described in Section 3.3: popularity-based approach *(Pop)*, user-based Collaborative Filtering with similarity computed with Tag usage *(CFUT)*, user-based Collaborative Filtering with similarity computed with tagged Item *(CFUI)* and FolkRank *(FR)*. In this chapter we also compare the performance of CSR with SR (Section 2.4.3) to demonstrate that it can achieve comparable recommendation performance while being more scalable.

5.4 Parameter Tuning

Implementing CSR requires the setting of a number of parameters. In this section, we report a number of experiments we conducted to select their values.

The first parameter refers to the threshold no_{high} used to distinguish leaders from followers. This parameter is set by studying the average tagging activity of users over the selected dataset and by selecting a value to elect as leaders the smallest set of users who collectively tagged most of the items. We experimented with two different thresholds: the first elects as leaders those users who have tagged more than 10 items (shortly called LM10), the second selects as leaders those users who have tagged more than 30 items (shortly called LM30). Table 5.2 reports, for each dataset, how many users are elected as leaders, how many items they have collectively tagged and how many tags they have collectively used with respect to the original dataset. Note that when using the threshold no_{high} equal to 30, less than 20% of the users are elected as leaders but they are still responsible for 94% of the tagged items. This confirms the fact that a small portion of users is responsible for the vast majority of the tagged items. We thus expect recall not to be severely affected when we restrict our attention to this small set of users only.

Dataset	Num. of Users/Leader	Num. of Items	Num. of Tags
Bibsonomy	1,360	23,649	11,668
Bibsonomy UM10	450 (33%)	23,103 (97%)	11,061 (94%)
Bibsonomy UM30	279 (20%)	22,386 (94%)	10,612 (90%)
CiteULike	2,484	7,310	3,137
CiteULike UM10	1,189 (47%)	7,291 (99%)	3,056 (97%)
CiteULike UM30	432 (17%)	7,116 (97%)	2,811 (89%)
MovieLens	1,270	3,400	2,237
MovieLens UM10	305 (24%)	3,334 (98%)	2,108 (94%)
MovieLens UM30	128 (10%)	3,278 (96%)	1,988 (88%)

Table 5.2: Clusters' characteristics

The second parameter affecting CSR is the number k of clusters, as well as the strategy used to initialise them. As explained in Section 4.4.2, the value of k has been chosen following the empirical rule of thumb described in [Mardia et al., 1979]. Moreover, to show that leaders were correctly classified by the Fuzzy c-Means algorithm, we plot in Figure 5.2 the similarity of each leader with the centroid of the cluster she is placed into. Note that, in the graph, leaders (x axis) are ordered according to the cluster they belong to and to their similarity value (y axis) within the centroid. The graph shows that most of the leaders have a high similarity with the centroid and that it never drops below 0.1. This indicates that leaders were correctly classified inside clusters.

To initialise the clusters, we experimented both with a *random point initialization* (i.e., each cluster centroid is chosen as a random point in the item space) and with a *real users initialization* (i.e., each cluster centroid is chosen as a real user, so that different centroids have no tagged items in common). For each strategy we measured the intra-group similarity (i.e., the similarity of each user with her centroid) as an indication of the clustering quality. Figure 5.1 illustrates an example of the measured intra-group similarity computed over the CiteULike LM30 dataset, with $k = 14$ and random-point initialisation, while Figure 5.2 illustrates the intra-group similarity with real-users initialisation. Note that, in the former case, most users have a similarity with the centroid of their cluster in the order of 0.001, while in the latter such value is never below 0.1. The use of real, non overlapping users' vectors for initialisation yielded better results (i.e., higher intra-group similarity). This is why we adopted this strategy throughout our experiments. The number of clusters we worked with (calculated as described in Section 4.4.2) is reported in Table 5.3, together with the number of iterations which were required to reach a stable state (clusters do not change anymore from one iteration to the next one).

We set the remaining parameters required by CSR as follows: query expansion was limited to a maximum of $5 * n$ tags, with n being the number of query tags. For query association,

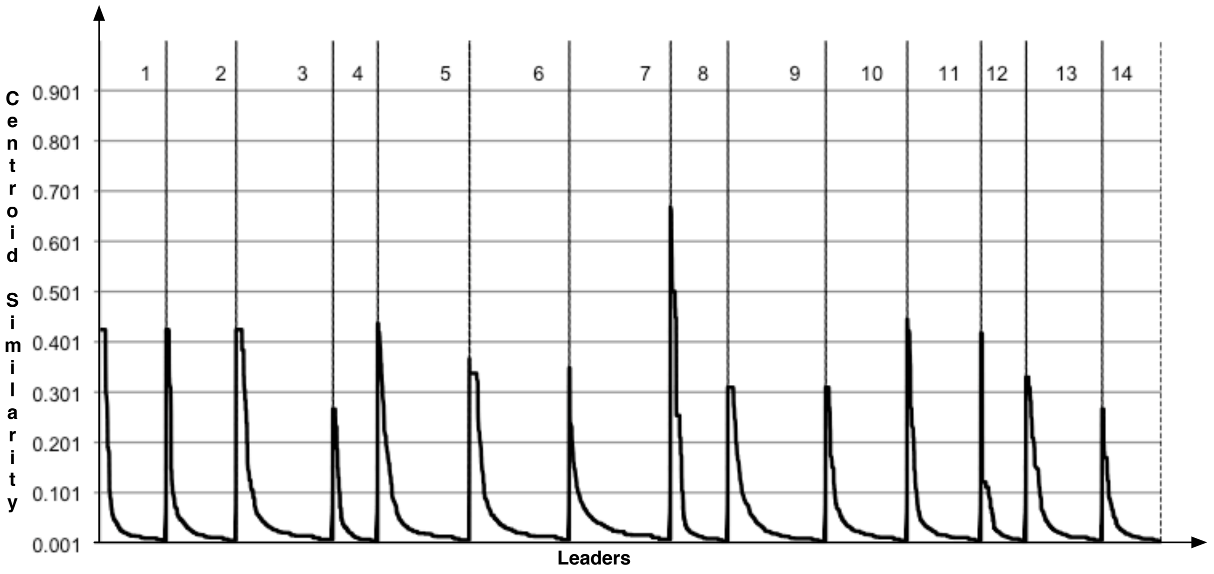

Figure 5.1: Clustering of leaders for CiteULike LM30 with random point initialization

the value of no_{low} required for a user not to be considered in the cold start region was set to 10. Finally, the α exponent used to mark differences between recommendations coming from clusters of different relevance was set to 5. As shown in the experiment results reported in Section 5.5, these settings for n, no_{low} and α guarantee that the performance of CSR are better than the benchmarks' one. A more fine-grained tuning of these parameter has been left as future work.

5.5 Results

We now present the results of our evaluation. We will focus on efficacy first, thus analysing precision/recall values of the various approaches on each of the three datasets under con-

Dataset	Num. of Leaders	Num. of Clusters	Num. of Iterations
Bibsonomy LM10	450	17	49
Bibsonomy LM30	279	13	14
CiteULike LM10	1189	26	12
CiteULike LM30	432	14	5
MovieLens LM10	305	13	7
MovieLens LM30	128	8	4

Table 5.3: Clustering features

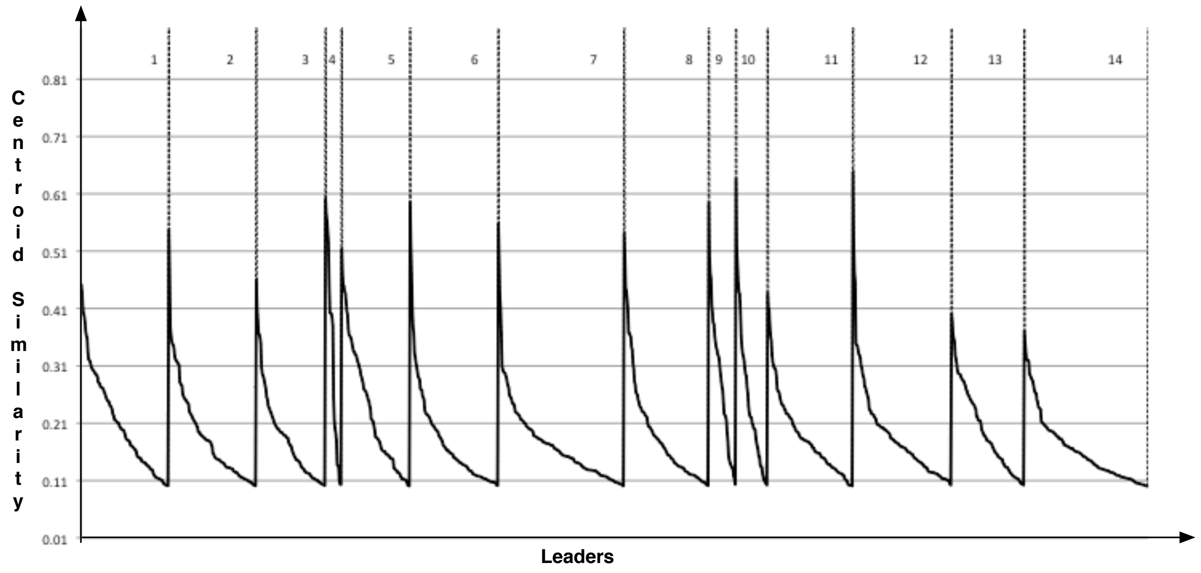

Figure 5.2: Clustering of leaders for CiteULike LM30 with real user initialization

sideration (Section 5.5.1). In particular, we consider two different settings. In the first one, shortly called CSRLM10, we select the group of *leaders* as the users who tagged more than 10 items, while in the second one, shortly called CSRLM30, we select the group of *leaders* as the users who tagged more than 30 items. As already pointed out in Section 5.4, these parameters are specific for the selected dataset and they cannot be generalized. They are set by studying the average tagging activity of users and by electing as leaders the smallest set of users who collectively tagged most of the items. As our approach has been devised to recommend items to new users, we present results divided in two groups: queries performed by active users (UM10), that is, those who have tagged at least 10 items in the training set, and queries performed by new users (UL10), that is, those who have tagged less than 10 items in the training set. These two groups have been selected only to emphasize the difference in performance between CSR and the considered benchmarks depending on the characteristic of the query user and searched item. The lower these values, the greater the difference between the algorithms. Note that to evaluate the performance of CSR we do not further split results according to the popularity of the searched item. We have already demonstrated that our proposed technique outperforms state-of-the-art ones in terms of efficacy. Our goal in this section is to demonstrate that CSR can achieve performance comparable to SR while improving scalability. In both settings, we discarded from the test set all queries for which the hidden item did not belong to the training set, since none of the implemented algorithms would have been able to answer such queries successfully. Table 5.4 reports, for each dataset, the number of test queries performed. We report in Section 5.5.1.3 a more general analysis of the precision/recall performance of CSR on the whole set of queries. We then evaluate the efficiency of CSR by analysing its computational complexity (Section 5.6).

Dataset	Total number of queries	UM10	UL10
Bibsonomy	1,342	753	589
CiteULike	4,575	3,156	1,419
MovieLens	2,038	769	1,269

Table 5.4: Number of test queries performed

5.5.1 Precision and Recall Computed on Each Activity Group

5.5.1.1 Bibsonomy

As Figures 5.3 and 5.4 illustrate, FolkRank is the best approach, both in terms of precision and recall, when dealing with active users. However, while the gain over standard CF approaches (e.g., CFUT and CFUI) is significant, the gain is much less with respect to CSR and SR. Moreover, as the recommendation list grows (x axis), this gain becomes lower. For example, when the recommendation list is cut at the top 50 results, FR has only a 14% improvement over CSRLM10 in terms of recall and 16% in terms of precision, while the improvement is 71% and 70% for precision and recall respectively, over CFUI.

If we consider new users instead (Figures 5.5 and 5.6) the situation is different and CSR (both CSRLM10 and CSRLM30) exhibits performance similar to SR and clearly outperforms all other approaches. In this case, FR looses as much as 47% in terms of precision and 44% in terms of recall with respect to CSRLM10 when looking at the top 50 results in the recommendation list. Note that CSR achieves performance comparable to that of the original SR, but it does so while leveraging information about a fraction of users only. These first results confirm the suitability of CSR to answer queries from new users (which constitute a non negligible fraction of all performed queries). Furthermore, CSR has performance that remains close to the best approach (i.e., FR) for active users too. In Section 5.6, we will extend the comparison between FR and CSR in terms of computational complexity, to prove that CSR is a better alternative to FR overall when scalability is a concern.

5.5.1.2 CiteULike

We now focus on the results obtained on the CiteULike dataset. As shown in Figures 5.7 and 5.8, CSR (CSRLM10 and CSRLM30), SR and FR all achieve very similar precision and recall values for active users (with (C)SR being slightly better than FR), outperforming both traditional CF approaches and simple Pop approaches. We now turn our attention

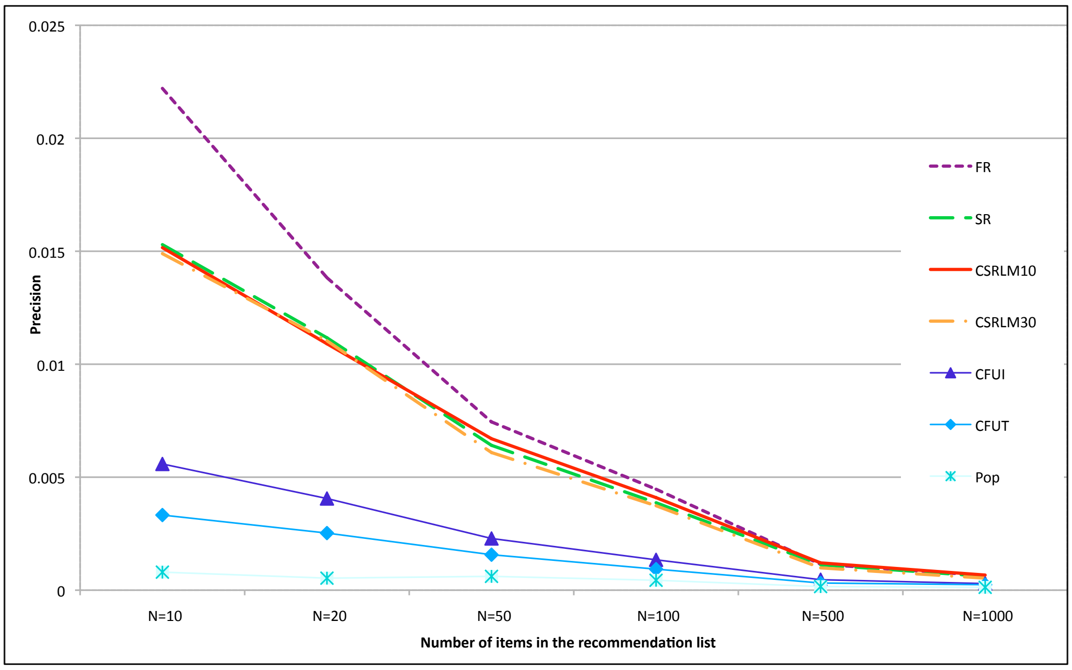

Figure 5.3: Precision for active users on Bibsonomy

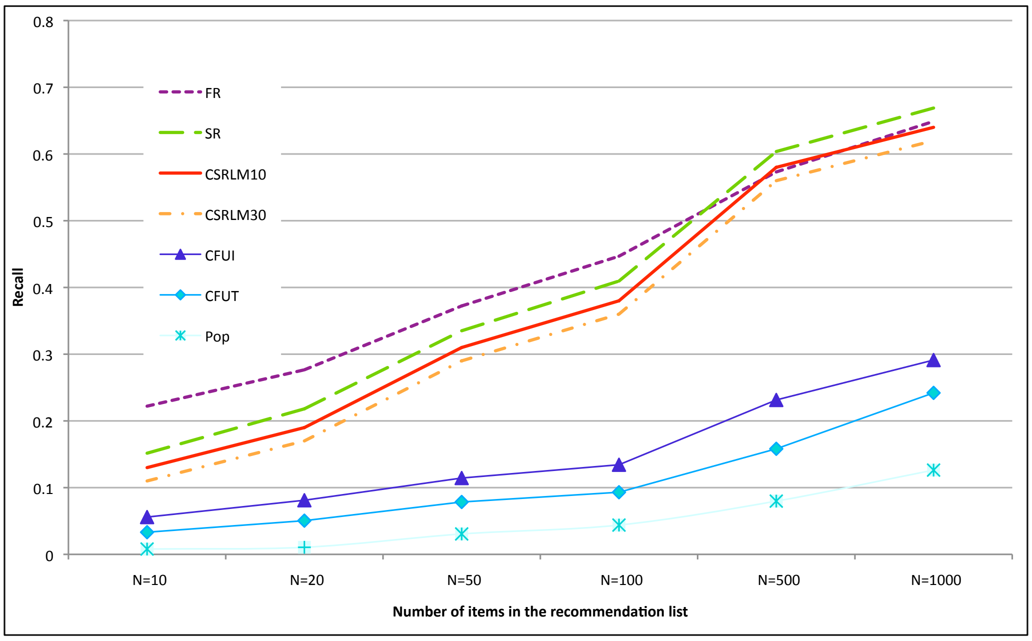

Figure 5.4: Recall for active users on Bibsonomy

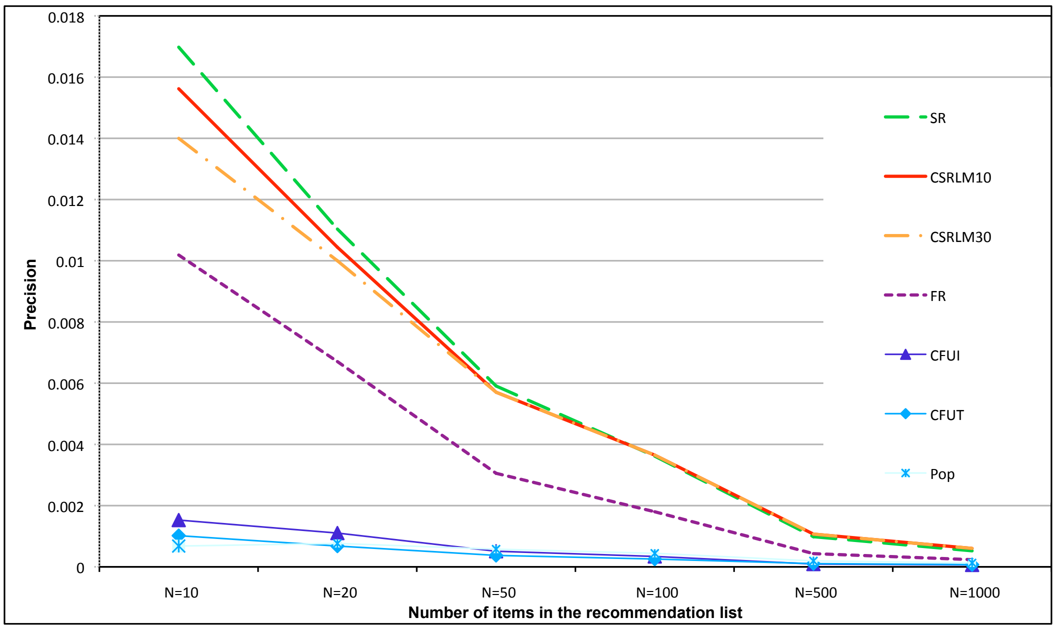

Figure 5.5: Precision for new users on Bibsonomy

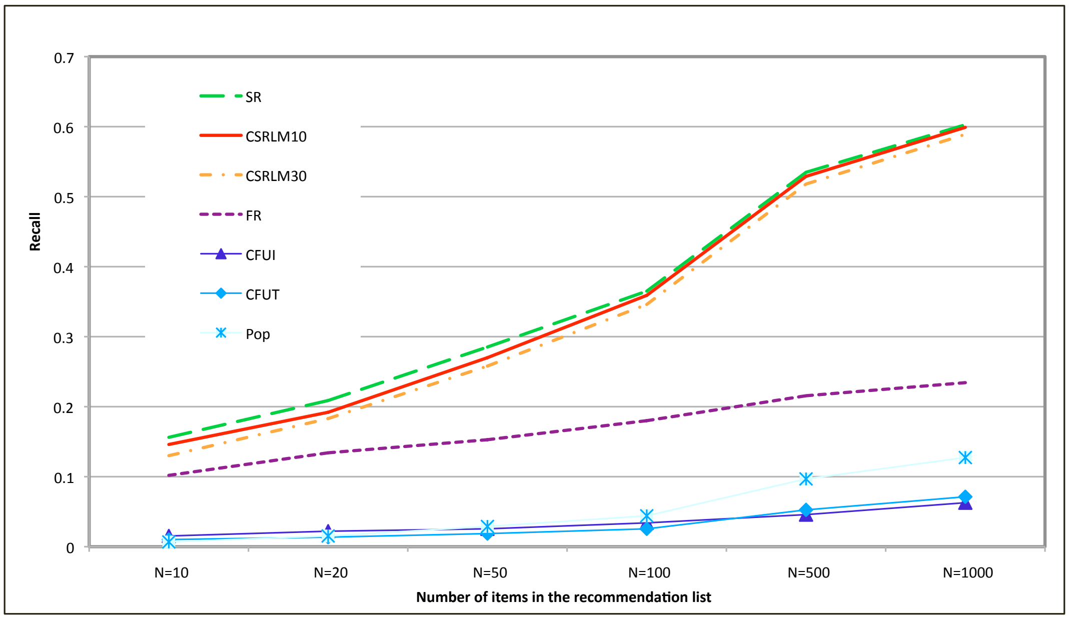

Figure 5.6: Recall for new users on Bibsonomy

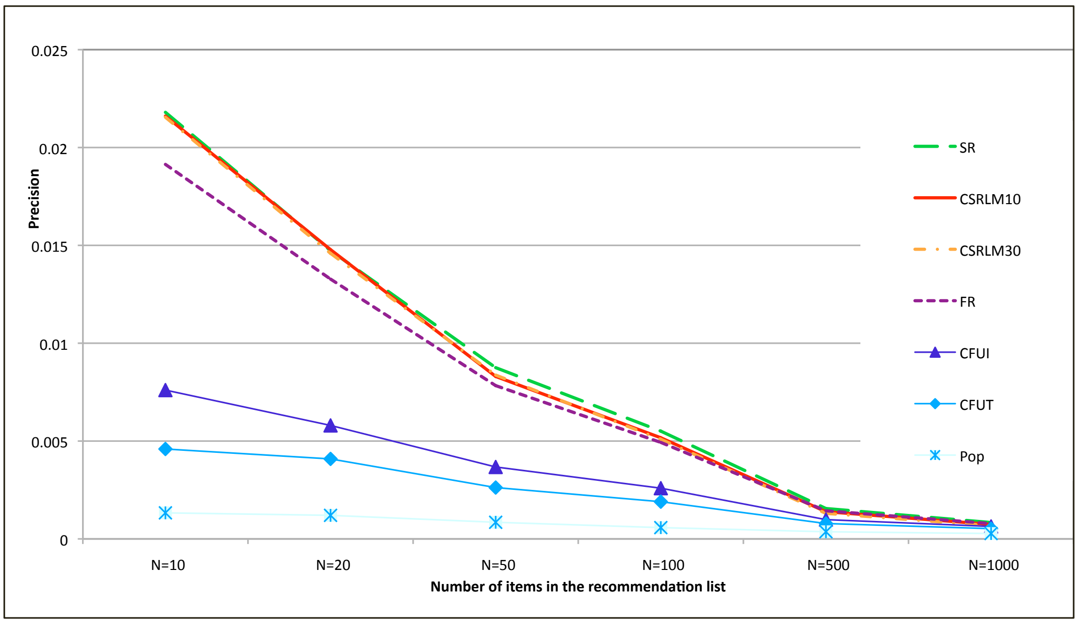

Figure 5.7: Precision for active users on CiteULike

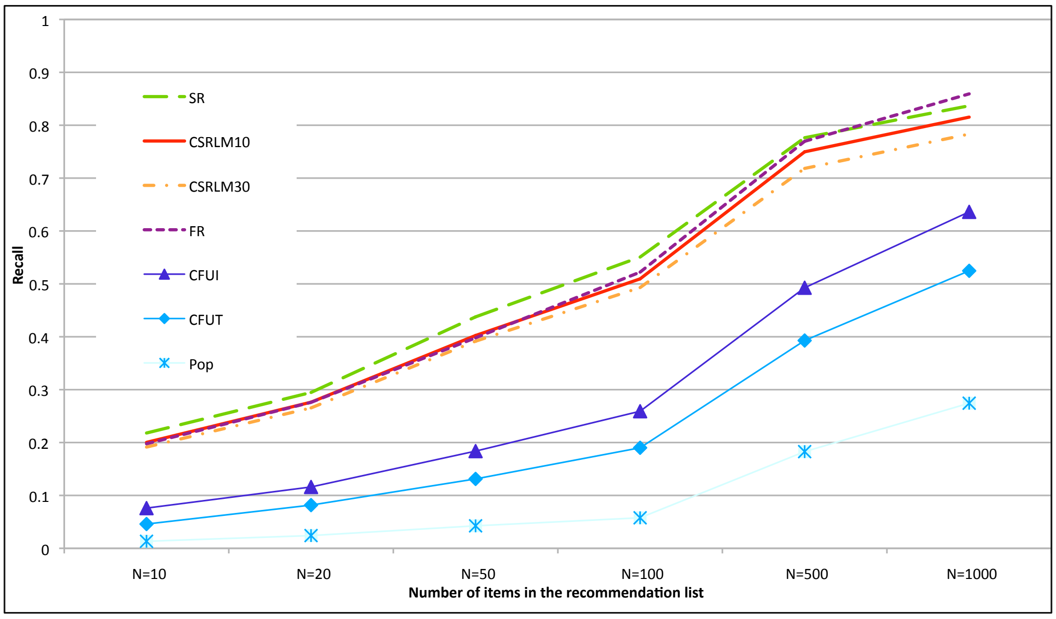

Figure 5.8: Recall for active users on CiteULike

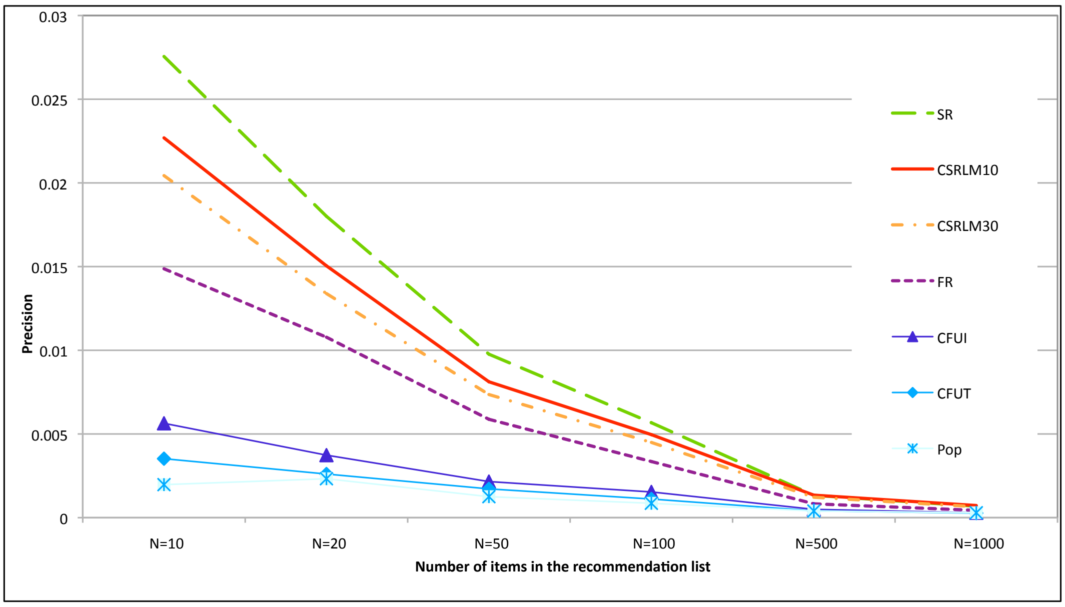

Figure 5.9: Precision for new users on CiteULike

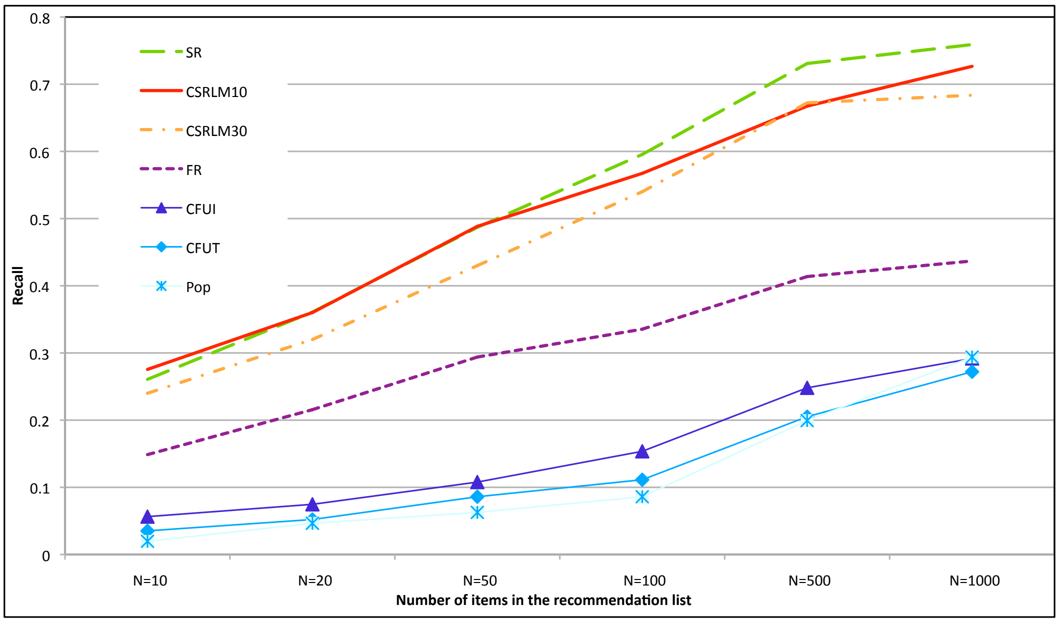

Figure 5.10: Recall for new users on CiteULike

to new users instead. As Figures 5.9 and 5.10 illustrate, FR cannot compute good recommendations, as too little information is available about the users. However, CSR exploits the little information available in the query and about leaders and obtains precision and recall values which are comparable to those of SR, and 28% and 40% respectively better than those obtained by FR (for recommendation lists of 50 elements).

5.5.1.3 MovieLens

The last dataset we analyse is MovieLens. As Figures 5.11 and 5.12 illustrate, FR, SR and CSR have very similar performance for active users. The same cannot be said for new users. As Figures 5.13 and 5.14 illustrate, SR and CSR achieve the best performance (with a gap of more than 85% for both precision and recall values and for recommendation lists of 50 elements).

Based on the experiments conducted over the Bibsonomy, CiteULike, and MovieLens datasets, we can thus conclude that SR and CSR are the most effective techniques to recommend items to new users, with a significant gain over other techniques. When considering active users, both SR and CSR have a performance (in terms of precision and recall) comparable to the best state-of-the-art approaches (FR). However, as we shall discuss next, the CSR has lower computational cost than FR and SR, and is thus the most suitable approach in scenarios where both efficacy and efficiency are important.

Precision and Recall Computed on the Whole Query Set

For completeness, we have analyzed precision and recall considering the overall set of performed query as a whole. As Figures 5.15-5.20 illustrate, CSR and SR achieve the best performance, both in terms of precision and recall. The performance gain for CSR both in terms of recall and precision goes from a 4% improvement over FR on the Bibsonomy dataset to 68% on the MovieLens dataset (for recommendation lists of 50 elements). These results again reinforce our previous conclusions: CSR outperforms state-of-the-art recommending techniques and achieves performance comparable with SR despite using only a fraction of the available data.

5.6 Complexity Analysis

In this section we focus on FR, SR and CSR, that is, the three approaches that exhibit higher efficacy, and analyse their computational complexity. When quantifying the computational cost of the different approaches, we distinguish between *offline* cost and *online* cost, as defined in Section 4.2. Table 5.5 reports the computational complexity of each approach.

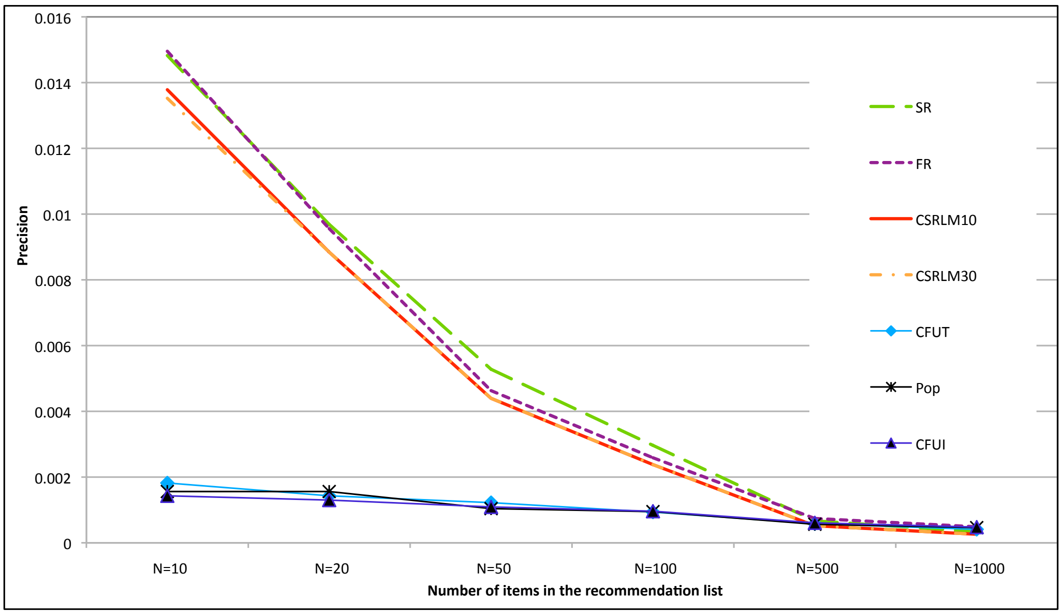

Figure 5.11: Precision for active users on MovieLens

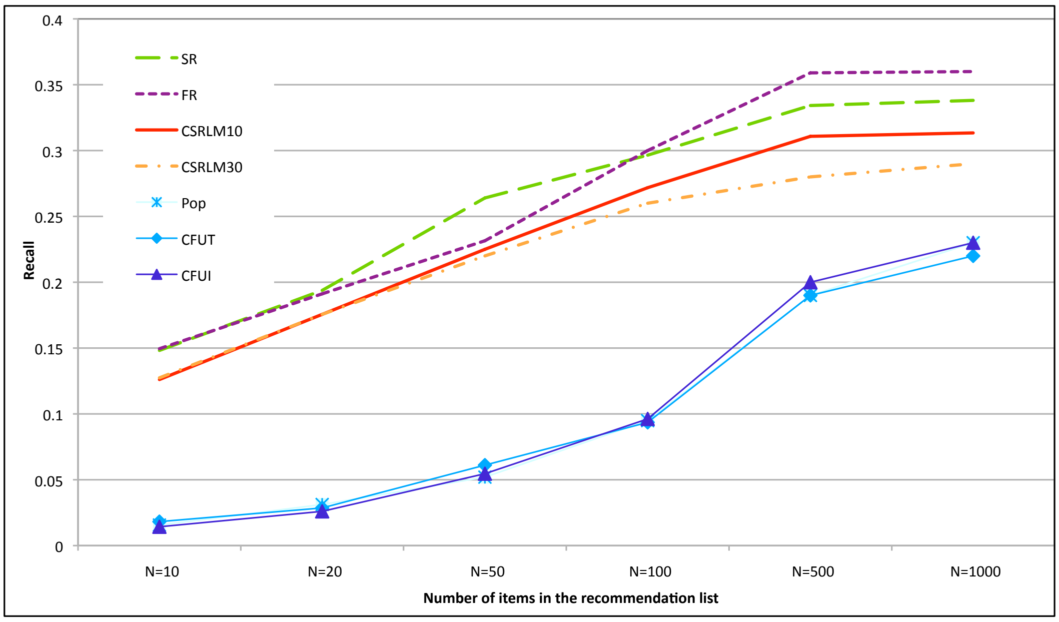

Figure 5.12: Recall for active users on MovieLens

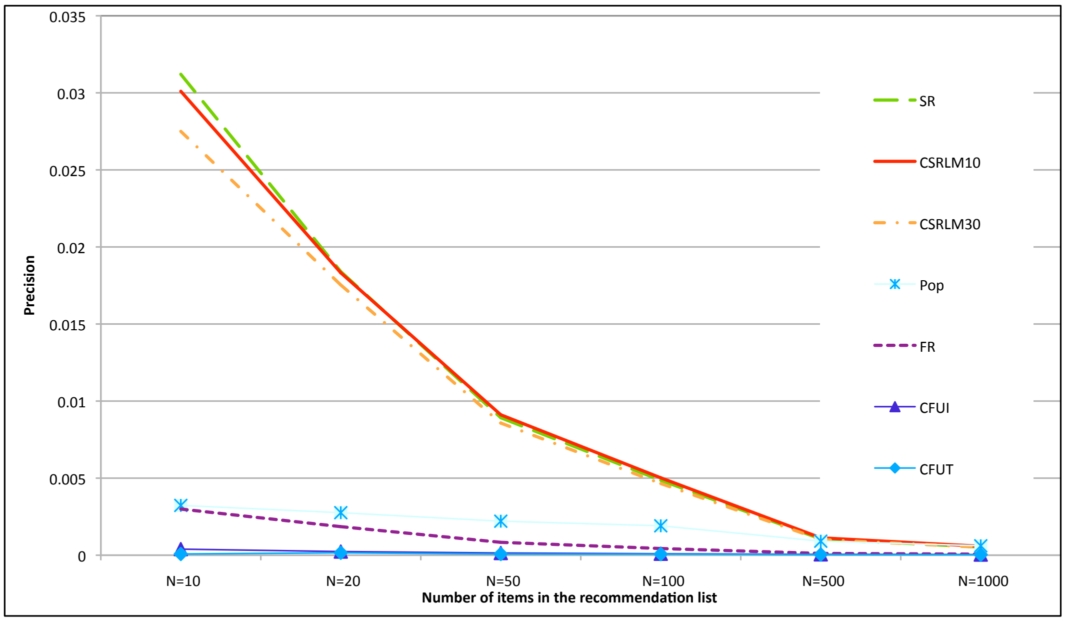

Figure 5.13: Precision for new users on MovieLens

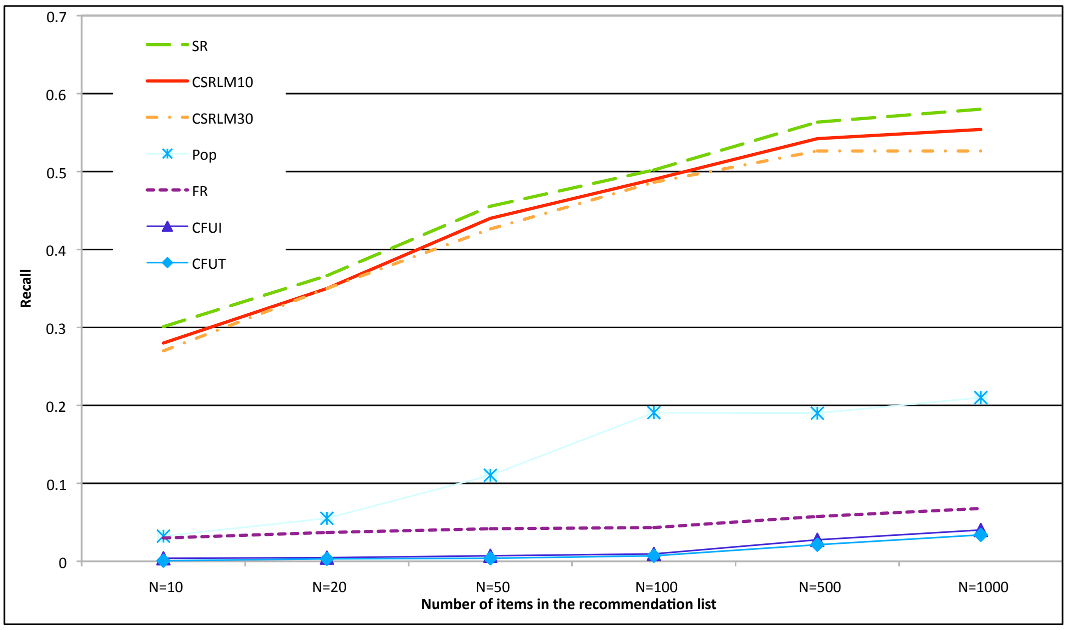

Figure 5.14: Recall for new users on MovieLens

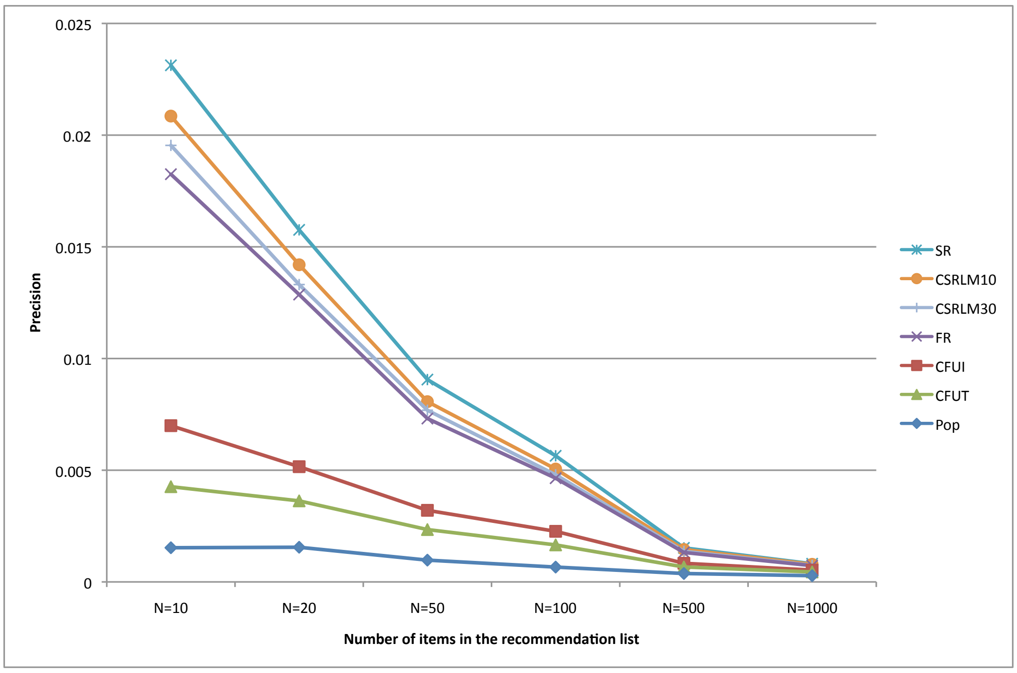

Figure 5.15: Overall precision on Bibsonomy

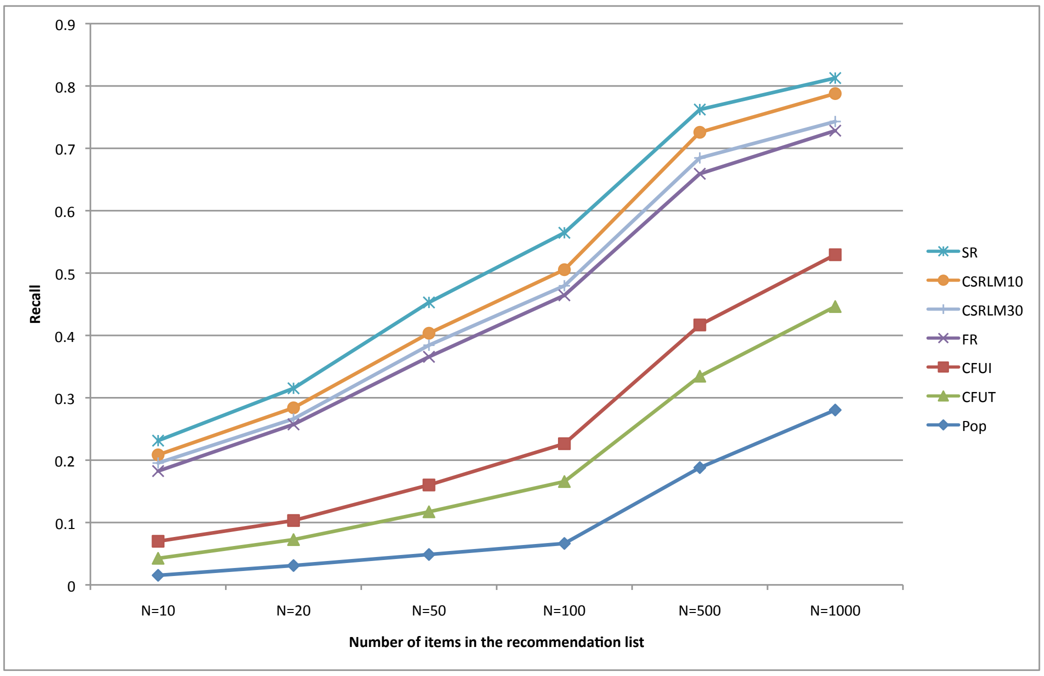

Figure 5.16: Overall recall on Bibsonomy

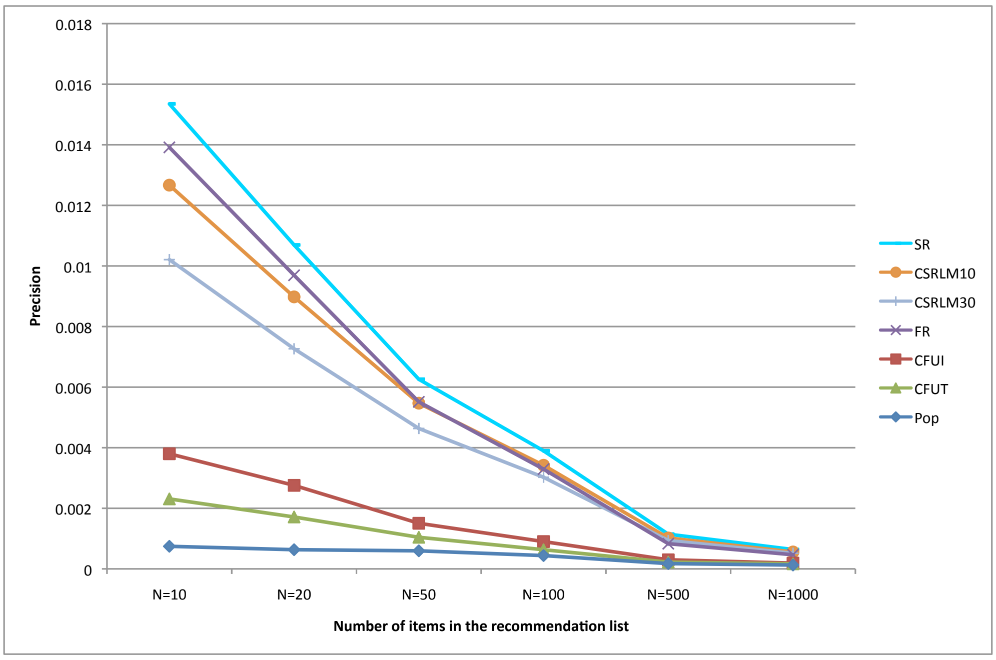

Figure 5.17: Overall precision on CiteULike

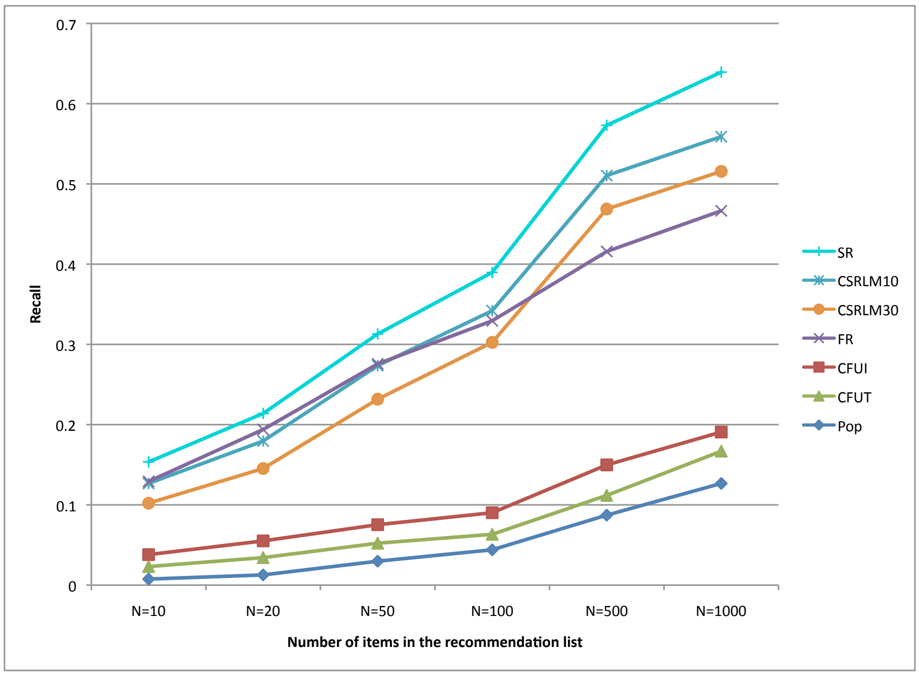

Figure 5.18: Overall recall on CiteULike

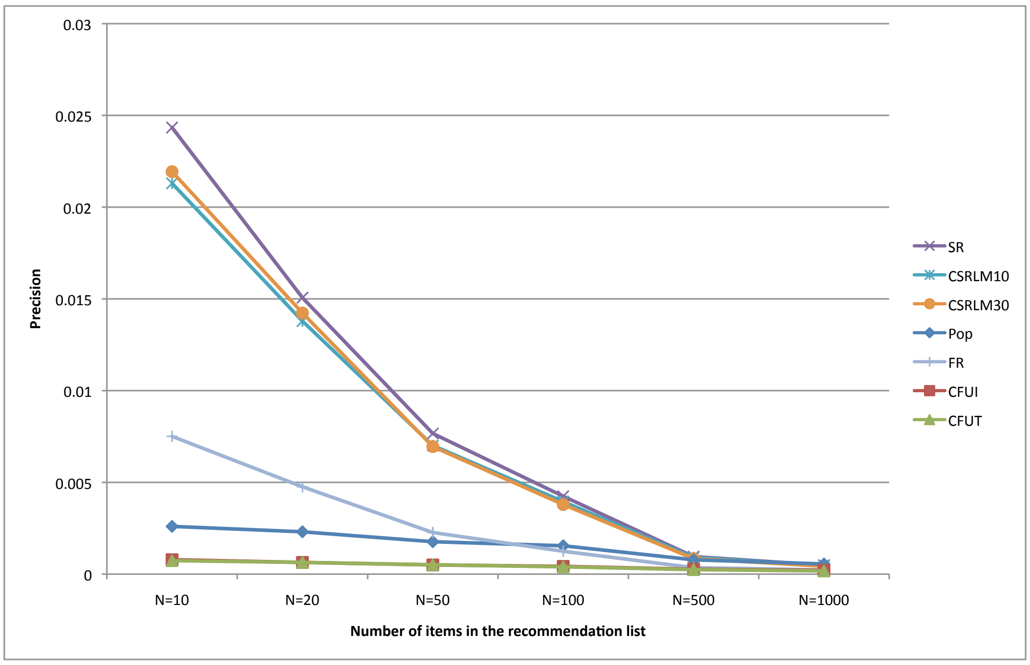

Figure 5.19: Overall precision on MovieLens

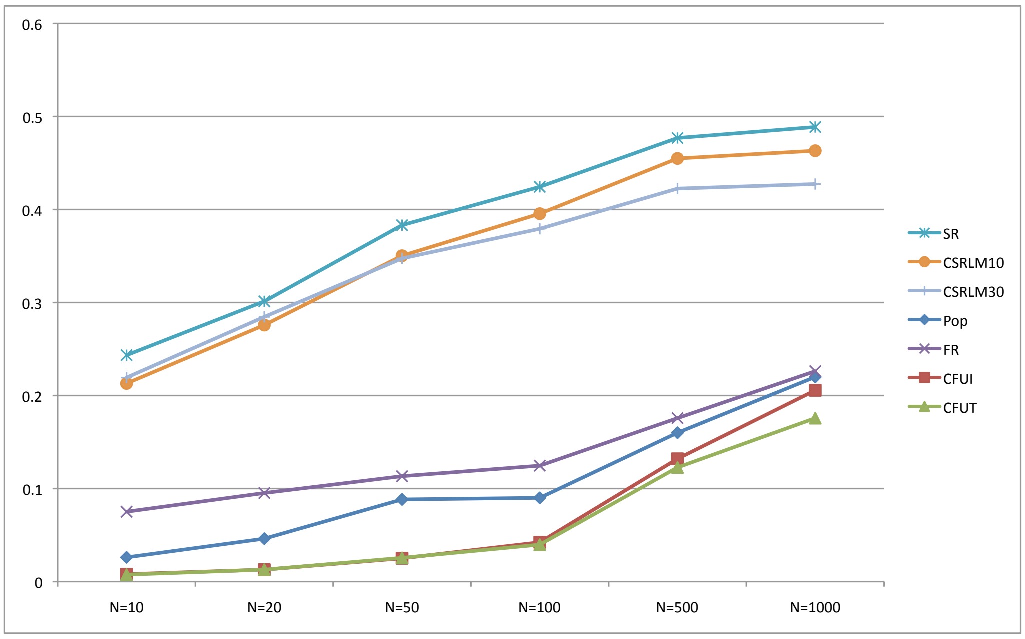

Figure 5.20: Overall recall on MovieLens

Approach	**Offline**	**Online (per query)**	**Offline actual**	**Online actual (all queries)**
FR	-	$O(iterations \times na)$	-	23M
SR	$O\left(\frac{nu\times(nu-1)}{2}+\frac{nt\times(nt-1)}{2}\right)$	$O(no \times nt)$	8M	10M
CSR	$O\left(i \times k \times nu + k \times \frac{nt'\times(nt-1)}{2}\right)$	$O(k \times no \times nt)$	1.5M	4M

Table 5.5: Computational complexity of FR, SR and CSR

We already described both *offline* and *online* costs of FR and SR (Section 4.2). We now focus on the computational complexity of CSR. CSR requires two offline computations: the execution of the Fuzzy C-Means algorithm to cluster leaders and the computation of the tags' similarity matrix for each cluster to perform the tag expansion. The former computation is linear in the number of leaders to cluster (nu in the worst case) as the number of iterations i required to converge and the number of clusters k are usually fixed in advance [Jain et al., 1999]. The latter computation involves a symmetric matrix, so the offline cost of CSR is $O(i\times k\times nu+k\times nt\times(nt-1)/2)$. The online computational complexity can be estimated as $O(k \times no \times nt)$ in the worst case where the query is associated to all clusters k, in which all no items have been tagged with all nt tags. Note that both the offline and online complexities are still quadratic. However, these complexities are just upper bounds as the real number of leaders that are clustered and of items and tags used is much smaller then the corresponding totals. To give an idea of the actual cost of each approach in a real scenario, we have counted the number of operations during both offline and online processes when answering all 4575 queries from the CiteULike LM30 dataset. The results are reported in the last two columns of Table 5.5. As shown, the online cost of FR is the highest amongst all approaches. In fact FR's main disadvantage is that it requires a complete computation of the Page Rank vector for each query, making it unsuitable for large datasets (as also confirmed by [Gemmell et al., 2009]). Both CSR and SR have a much smaller computational cost overall (offline + online).

We now take a closer look at CSR and SR. The offline cost of CSR is one order of magnitude smaller than that of SR. This is because the set of *leaders* is much smaller than the whole set of users, so the cost of clustering them is much smaller than computing all users' similarities. For example, in CiteULike LM30 there are only 432 leaders, as opposed to 2484 users overall. The cost to cluster leaders is the cost of 30K computations (with $k = 14$ clusters and $i = 5$ iterations to converge), while the cost to compute all users' similarities is 3M computations. Moreover, each cluster contains around 450 tags, as opposed to the 3137 tags overall. Even if a separate tags' similarity matrix must be computed for each cluster, the cost of computing tags' similarities for all of them (1.4M computations) is smaller than the cost of computing the complete tags' similarity matrix maintained by SR (5M computations). This results in an overall offline cost for CSR (30K for the clustering phase + 1.4M for tags' similarity matrices for all clusters = 1.4

M) smaller than that of SR (3M for computing the users' similarity matrix + 5M for the tags' similarity matrix = 8M). Furthermore, the online cost of CSR is half that of SR. SR performs each query with an expanded tag set of roughly 10 tags, each associated with 200 items (10M computations for all 4575 queries). CSR instead performs each query associating it with 4 clusters on average and using an expanded tag set of 5 tags, each associated with 40 items (4M computations for all 4575 queries).

The neat reduction in the offline cost of CSR also means that the data structures can be re-computed more often, thus again increasing accuracy without compromising scalability. Note that frequent updates are of utter importance in rapidly growing settings and especially for new users, where one update can make the difference between knowing a little about her preferences (her first few bookmarks) and knowing nothing at all. We can thus conclude that, when dealing with rapidly growing scenarios both in terms of users and items, CSR represents the most effective and efficient approach. In fact CSR can compute accurate recommendations for both old and new users and include both mainstream and non-mainstream items with only a small computational overhead.

Chapter 6

Adaptive Update

In the previous chapter we have presented Clustered Social Ranking (CSR), a *scalable* recommendation technique addressing the *user* and *item cold start problems*. Our evaluation, conducted on three different datasets (namely CiteULike, Bibsonomy and MovieLens) demonstrated that CSR achieves high efficacy with low computational cost. The evaluation we performed is standard across the literature. However, it assumes 1) that all recommended items are included in the training set, 2) that all tests are run at once over this training set and 3) that all used data structures are up-to-date. In practice, real datasets grow continuously over time and data structures must be regularly updated. Frequent updates result in better performance but also higher cost while unfrequent updates result into low costs but also lower performance. As such, the previous evaluation cannot be considered realistic. Recommendation algorithms must therefore cope with stale data in the period between two updates. Deciding whether a system update would be worth its cost is the main issue system administrators must face. In this chapter we describe a temporal analysis of the performance of CSR and show what effects stale data can have on the recommendation process (Section 6.1). We then define an empirical technique to dynamically decide when to perform the next data update (Section 6.2). Finally, we evaluate the performance of CSR when our proposed updating technique is applied (Sections 6.3 and 6.4).

6.1 Analysis of the Problem Space

As discussed in [Gunawardana and Shani, 2010], the research community has usually evaluated recommender systems according to the following methodology. The available dataset is first divided into two subsets: a training set and a test set. The training set is used to build the data structures representing the system while the test set is used to test the performance of the solution. This means that the training set and its corresponding data structures remain unchanged for the whole duration of the tests. In other words the resulting evaluation is based on a system that never changes. Unfortunately, this assumption is not realistic. In fact, real-world systems need to cope with growing sets of both users and items and must thus perform periodic updates of their data structures. Companies are usually reluctant to divulge the details of their updating strategies as they are core to their business. Regularly updating the data structures used during the recommendation process is of utter importance for recommendation efficacy. For example, new items inserted after an update could not be recommended while new bookmarks could not be taken into account when calculating recommendations. However, it also is an expensive process that takes both time and computational resources. To limit the costs, data updates are therefore performed only at fixed time intervals. In [Mull, 2006] the authors state for example that their system is updated once a week. However, while immediately after an update recommendations are based on all available data, at the end of the interval recommendations will be based on stale data. In particular, the more information has been inserted during the interval, the more severe its impact will be on performance. This is even more true if new users or new items are inserted, that cannot receive recommendation or be recommended at all, respectively, until the next update.

To evaluate the effect of data growth on the performance of recommender systems, we perform an experiment by focusing on the Bibsonomy dataset. We consider a set of bookmarks inserted between February 1995 and June 2009 and we preprocess them so to consider only users, items and tags occuring in at least p posts/bookmarks, with $p = 2$ (Section 3.2). We then analyse the growth of the dataset over time and plot for each month the number of users, items and tags belonging to the system (Figure 6.1). The analysis points out that there exists an initial period (February 1995-February 2005) after the system creation in which users, items and tags grow slowly over time following similar patterns. This slow growth is typical of a system in its early stages, e.g., when its first prototype version has just been released and only a few users know and use it. However, as the system gains popularity, users, items and tags grow faster and with very different patterns. As expected, bookmarks have the fastest growth followed by items, tags and users. New bookmarks in fact most probably contain an existing user tagging a new item with existing tags. For this reason, there are time periods when items grow faster than both users and tags. There are however also time periods when tags grow faster, especially at the initial stages of the system when the vocabulary of tags is still being constructed with new terms. We hypothesize that the different speeds of growth of items and tags have different impact on the recommendation efficacy.

To verify what is the influence of different data growths on recommendation efficacy and how performance can be improved by data updates we performed the experiment depicted in Figure 6.2. We consider all new bookmarks inserted in the dataset during a periods of one month T. We build two different training sets: a first training set TS_1 that contains all bookmarks added in the system before T and a second training set TS_2 that contains

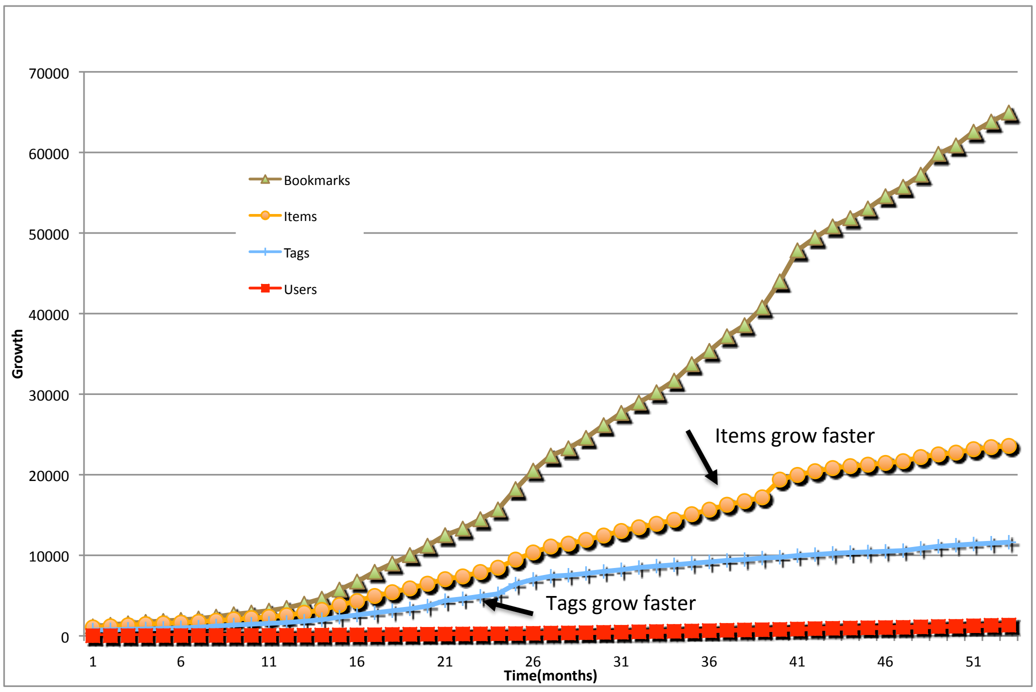

Figure 6.1: Growth of users, items and tags in the Bibsonomy dataset

Time period	User growth	Tag growth	Item growth
T_1: October 2006-November 2006	6%	17%	2%
T_2: January 2008-February 2008	6%	1%	11%
T_3: June 2007-July 2007	6%	2%	3%

Table 6.1: Feature of data snapshots

all bookmarks added in the system until the end of T. Note that with this construction $TS_1 \subseteq TS_2$. We then train two different instances of CSR, namely CSR_1 and CSR_2 using TS_1 and TS_2, respectively. We expect a loss in performance when the system is trained only with TS_1 that depends on both the intensity of the users' tagging activity during T and on the number of queries related to items and tags inserted during T.

Note that in this experiments we only focus on the impact of data growth over the recommendation accuracy provided by CSR for Bibsonomy dataset for time-related issue. This choice has been done considering that CSR has been developed with the goal of addressing the scalability problem that affects the vast majority of recommendation algorithms including SR. By grouping users into clusters, CSR alleviates the scalability problem but it requires the data in the clusters to be kept up to date to ensure high recommendation efficacy. Evaluating the impact of data growth over time over SR and over other datasets has been left for future work. To understand how the users' tagging activity affects the performance, consider a set L of *leaders* and a set F of *followers* at the beginning of time period T. If during T some of the users in F become *leaders*, CSR_1 will end up in recommending items based on a set of data that does not reflect the actual state of the dataset. Furthermore, the performance of CSR_1 is also affected by the tagging activity of the users already in L. All of their bookmarks, even the newest, should in fact be taken into consideration when calculating recommendations. To minimise the information loss, CSR_1 considers all the new items inserted by users in L even if it is not fully re-trained (i.e., clusters are not recomputed). This is the case also in real-world rating-based recommender systems, where users' profiles are constantly updated as new information is inserted. In our case we only consider new bookmarks by users in L because our solution only rely on leaders' recommendations. This low cost method ensures that new items inserted by users in L after the last update can still be recommended by CSR_1. However, since the tags' similarity matrices are not recomputed, every time a user performs a query specifying tags added after the beginning of T, CSR_1 will not be able to expand them. We consider the above setup for three different periods, as shown in Figure 6.3. The three periods were chosen so that during T_1 tags grew faster than items, during T_2 items grew faster than tags and during T_3 items and tags grew similarly and slow. Table 6.1 reports the three subsets of bookmarks and their corresponding growth rates. (T_1, T_2 and T_3)

Let us call the set of queries related to items and tags inserted during a time period T_i as Q_i^{new} and the set of queries related to items and tags inserted before T_i as Q_i^{old}. To evaluate how the loss in performance depends on Q_i^{new} we build for each time period T_i three different test sets of 1500 queries. In particular the first test set, shortly called *25%-75%*, contains 25% of queries from Q_i^{new} and 75% of queries from Q_i^{old}. The second test set, shortly called *50%-50%*, contains 50% of queries from Q_i^{new} and 50% of queries from Q_i^{old}. Finally, the third test set, shortly called *75%-25%*, contains 75% of queries from Q_i^{new} and 25% of queries from Q_i^{old}. For each T_i we then run both CSR_{i1} and CSR_{i2} on all three available test sets. We then evaluate the difference in performance (i.e., the difference in precision/recall) between the two CSR_{i1} and CSR_{i2}.

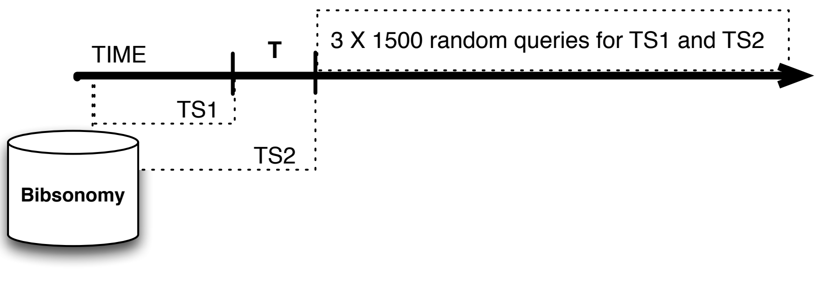

Figure 6.2: Experiment setup

Table 6.2 reports for each time period and test set the loss in precision and recall that we

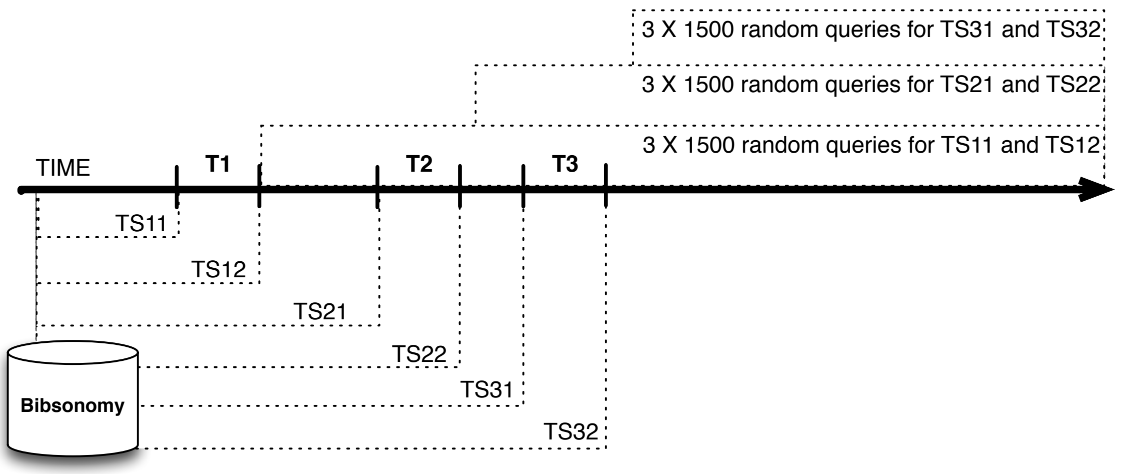
3 X 1500 random queries for TS31 and TS32
3 X 1500 random queries for TS21 and TS22
3 X 1500 random queries for TS11 and TS12
TIME
T1
T2
T3
TS11
TS12
TS21
TS22
TS31
TS32
Bibsonomy

Figure 6.3: Experiment setup

Time period	**25%-75%**	**50%-50%**	**75%-25%**
T_1	24%	32%	45%
T_2	12%	20%	23%
T_3	3%	10%	14%

Table 6.2: Precision/recall loss after 1 month

experience when using CSR_{i1} rather then CSR_{i2}. For simplicity the table contains only one percentage value for each setting. The loss in precision and recall are in fact almost identical. This is due to the fact that in these experiments only one item at a time is considered relevant for the user when performing the tests (see Section 3.2). The results confirm our previous considerations. First, the performance loss is affected by the amount of query from Q_i^{new}. The larger is the number of queries related to new items and tags, the higher is the precision/recall loss. Second, the performance loss is also affected by the intensity of the users' tagging activity. In fact, we observe a higher precision/recall loss for time periods T_1 and T_2, during which tags and items grow faster. It is also interesting to note that the performance loss for time period T_1 (during which tags grow faster) is twice as high as the one for T_2 (during which items grow faster). As previously discussed, the profiles of all users in L contain all the new items introduced by *leaders* during T_i, so that these items can still be recommended even by CSR_{i1}. Only items introduced by users in F are completely discarded and this cause a performance loss between 12% and 23%. However all new tags introduced by both users in L and F are not included in the tags' similarity matrices since CSR_{i1} is not fully re-trained. This causes an higher performance loss ranging from 24% to 45% that can be explained as a consequence of the query expansion that is at the base of (C)SR. In fact, the expansion cannot be performed properly if tags similar to the query ones (that are new) cannot be found.

6.2 A New Update Methodology

The experimental results reported in Section 6.1 show the importance of data updates to guarantee effective recommendations. However, because of its high cost, it is also prohibitive to perform updates every hour or every day. It is therefore necessary to balance the tradeoff between desired recommendation efficacy and number of updates. Existing recommender systems perform updates periodically (every week or fortnight). However, the results reported in Table 6.2 show that items and tags do not grow linearly over time and that performance is affected by data growth (Table 6.2). During specific periods of time the dataset's growth might be irrelevant and an update would not be worth its cost. During other periods of time instead, the dataset's growth might be such that even an earlier update (i.e., performed before the current period's end) would be highly beneficial. Our goal is to define a new adaptive technique that constantly monitors the dataset's growth and re-trains the system every time the gain in recommendation efficacy would outweight the update's cost. Since cost and gain depend on the specific system, we let its administrator decide by choosing a threshold ut such that the system is retrained every time its growth since the last update is above ut, which would cause a predictable performance loss. In other words, we estimate the loss in performance by looking at how much the dataset has grown and we let the system administrator decide when to perform an update on the base of the parameter ut only.

Since the performance of the recommendation algorithm is particularly affected by the growth of items and tags, our technique keeps track of the sum $Grow_{\%}$ of the growth percentages of new items and tags inserted *since the last update*. Whenever $Grow_{\%}$ exceeds a specific threshold ut, the technique autonomously decides to perform a system update of all the data structures. In this situation, performing a data update is considered necessary to guarantee effective recommendations. The value of ut can be tuned by the system administrator depending both on the level of recommendation efficacy that must be provided and on the computational overhead that the system can afford. Small values of ut guarantee better efficacy but higher costs, while high values of ut ensure a cheaper but less effective computation. Note that we decided to use a single threshold ut instead of two separate thresholds for the percentage growth of new items and tags to limit the parameter tuning required by the system administrator.

6.3 Experiment Setup

We now evaluate the performance of the proposed adaptive update methodology. In particular, we compare the performance of CSR when three different update strategies are applied: *Monthly*, that performs periodic updates at the beginning of each month, *Adaptive*, that performs updates depending on $Grow_{\%}$, *Weekly*, that performs periodic update at the beginning of each week. Our goal is to demonstrate that the *Adaptive* strategy has a computational overhead which is comparable with the *Monthly* strategy (the cheapest approach but also the least effective) and a recommendation efficacy comparable with the *Weekly* strategy (the most expensive approach but also the most effective). Note that this setting is suitable for the specific dataset we consider. If the dataset grew faster we could compare the *Adaptive* strategy with a *Daily* and a *Weekly* strategy. We designed an experiment to find the cumulative error of CSR when using each of the three strategies.

We first perform a preparation step where we evaluate the effect of $Grow_{\%}$ on the recommendation process when the data structures are not updated. We do this to estimate a reference value indicating what is the performance loss following a specific dataset growth. Figure 6.4 depicts the prediction error, computed as the percentage loss in precision/recall when the value of $Grow_{\%}$ ranges between 1% and 10%, assuming that an update would be performed in any case with values of $Grow_{\%}$ greater than 10% (even with the *Monthly* update strategy the dataset never grows over 10% before an update). The depicted values where obtained with the following experiment whose setup is similar to that presented in Section 6.1. We choose ten different time periods (of different length) T_i with $1 < i < 10$. Each period is included between two different updates up_{i1} and up_{i2} and is such that the data growth $Grow_{\%i}$ during each period T_i is equal to i%. We then evaluate the error in prediction of CSR as the loss in precision/recall when the training is performed with the data updated at the beginning of T_i with respect to when the training is performed at the end of T_i. To evaluate the error, we create three different test sets for each T_i (so to obtain 30 test sets in total): *25%-75%*, *50%-50%*,*75%-25%*. Each test subset contains a random set of 1500 queries related to items and tags inserted after the end of T_i, as described in Section 6.1.

Figure 6.4 reports, for each of the three test sets, the error in prediction that CSR experiences for values of $Grow_{\%}$ ranging from 1% to 10%. The growth is represented on the X axis of the diagram, while the error is represented on the Y axis. The graph shows that the error in prediction grows almost linearly with $Grow_{\%}$. Moreover, the error increases depending on the percentage of query related to new items and tags inserted during T, confirming our previous observations. This error table can be used by the system administrator to decide which threshold value can better guarantee a certain level of recommendation efficacy.

After this preparation step, we consider an 81-week long period between October 2006 and March 2008 (so to include T_1, T_2 and T_3 previously considered) and we measure after each week the value of $Grow_\%$ for the *Adaptive (A)*, *Weekly (W)* and *Monthly (M)* strategies. Remember that $Grow_\%$ measures the growth of the system *with respect to the last update*. Therefore, since each strategy performs updates at different times, we can calculate a different $Grow_\%$ for each strategy. Each week will therefore be associated with three different values of $Grow_\%$, one for each update strategy that can be used, and thus also with a prediction error (found during the preparation step and depicted in Figure 6.4). We then calculate for each strategy the cumulative prediction error of CSR as the sum of all errors in prediction up to the considered week. The results are reported in Figure 6.5, 6.6 and 6.7. We also evaluate the number of updates that each strategy perform during the whole considered period as an indication of the computational overhead of each strategy. The results of the experiment are discussed in the following section.

6.4 Results

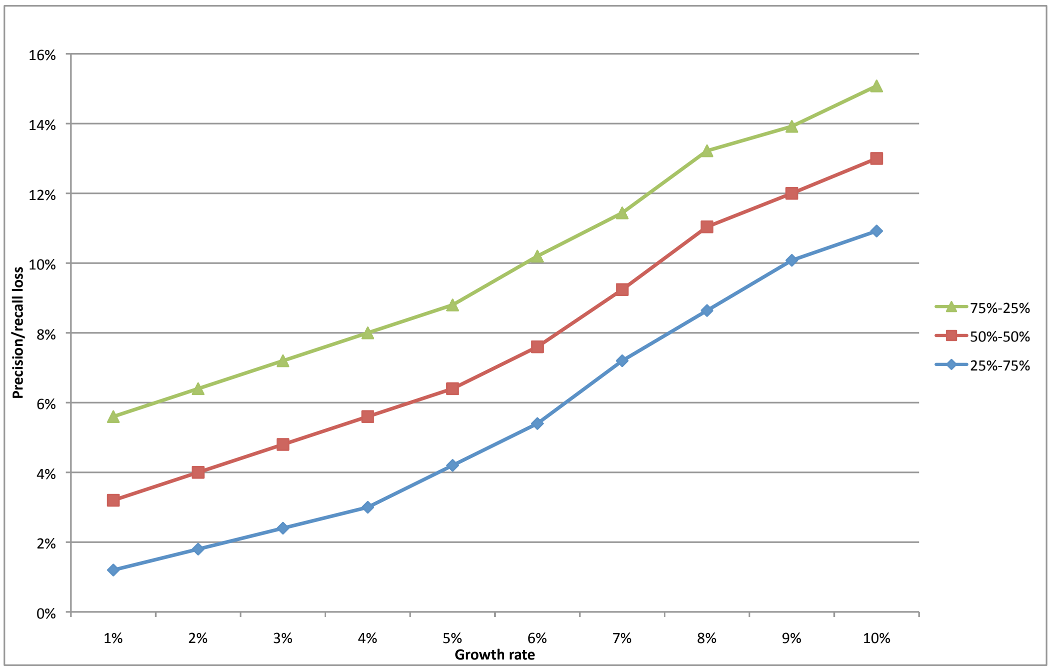

Figure 6.4: Error in prediction on Bibsonomy for different data growth

Figures 6.5, 6.6 and 6.7 show for each of the three types of test sets considered (*25%-75%*, *50%-50%*, *75%-25%*) the cumulative prediction error made by CSR if updated according to the *Adaptive*, *Weekly* and *Monthly* strategies. In particular, we consider two different threshold for the *Adaptive* technique, so that a data update is performed every time the value of $Grow_\%$ exceeds 2% (*Adaptive 2%*) and 5% (*Adaptive 5%*). Table 6.3 reports instead the number of updates computed by each updating strategies over the considered time period (81 weeks overall).

Note that the values for the thresholds are not unique and need to be tuned by system administrators according to the features of the considered dataset, as described in Section 6.2. We decided to evaluate system performance when these thresholds were set to 2% and 5% since these values enable us to show that the recommendation efficacy for both *Adaptive 2%* and *Adaptive 5%* with *25%-75%* queries is comparable to the *Weekly* strategy, and it greatly outperforms the *Monthly* strategy, as reported in Figure 6.5. Furthermore, the difference in performance grows for *50%-50%* and *75%-25%* queries, as shown by Figures 6.6 and 6.7, respectively. Note that Figure 6.7 shows that both *Adaptive 2%* and *Adaptive 5%* have much worse performance than the *Weekly* strategy when most of the queries refer to new items. This suggests that our strategy could be improved by letting the *ut* parameter vary according to the predicted incoming queries. The variation of *ut* could be decided on the basis of the most recently received queries. If it were possible to predict that most of the queries that will be issued in the future would refer to new items (i.e., the set of future queries would be of type *75%-25%*), then *ut* could be decreased. Conversely, *ut* could be increased if most of the queries that will be issued in the future would refer to old items. We leave further research on this idea for future work.

However, it is important to consider that *Adaptive 2%* performs only 29 updates with respect to the 80 performed by the *Weekly* strategy. This means that it would be possible to further reduce the threshold value *ut* to improve the recommendation efficacy while still keeping a low cost.

Technique	**Number of updates**
Weekly	80
Adaptive 2%	29
Adaptive 5%	13
Monthly	18

Table 6.3: Number of system updates performed by each strategy

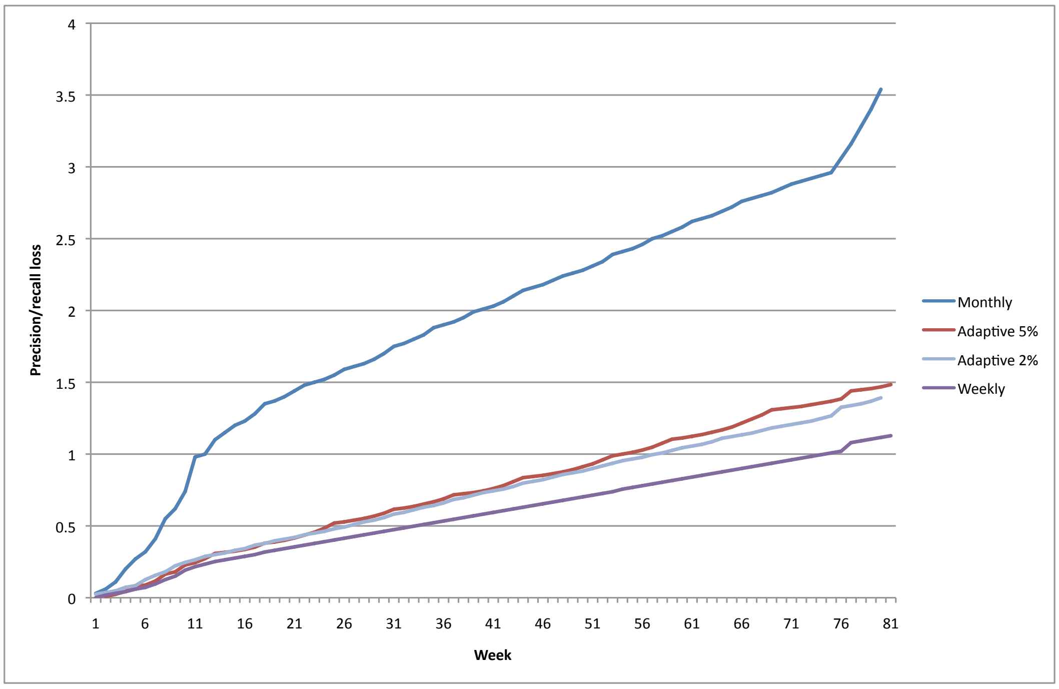

Figure 6.5: Cumulative error on Bibsonomy computed for 25%-75% test set

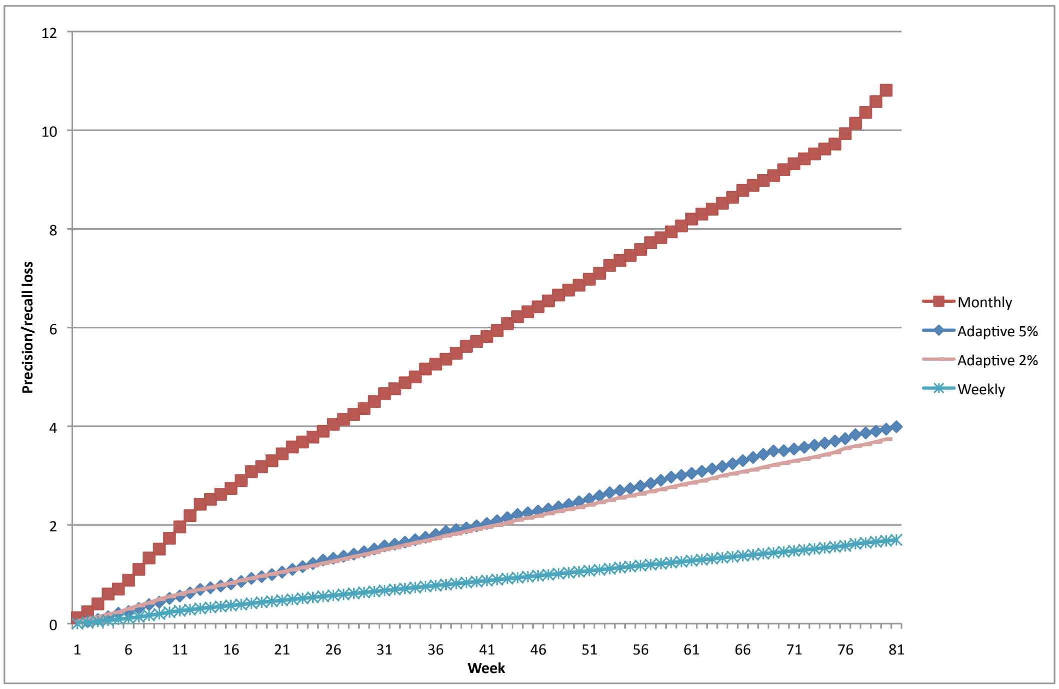

Figure 6.6: Cumulative error on Bibsonomy computed for 50%-50% test set

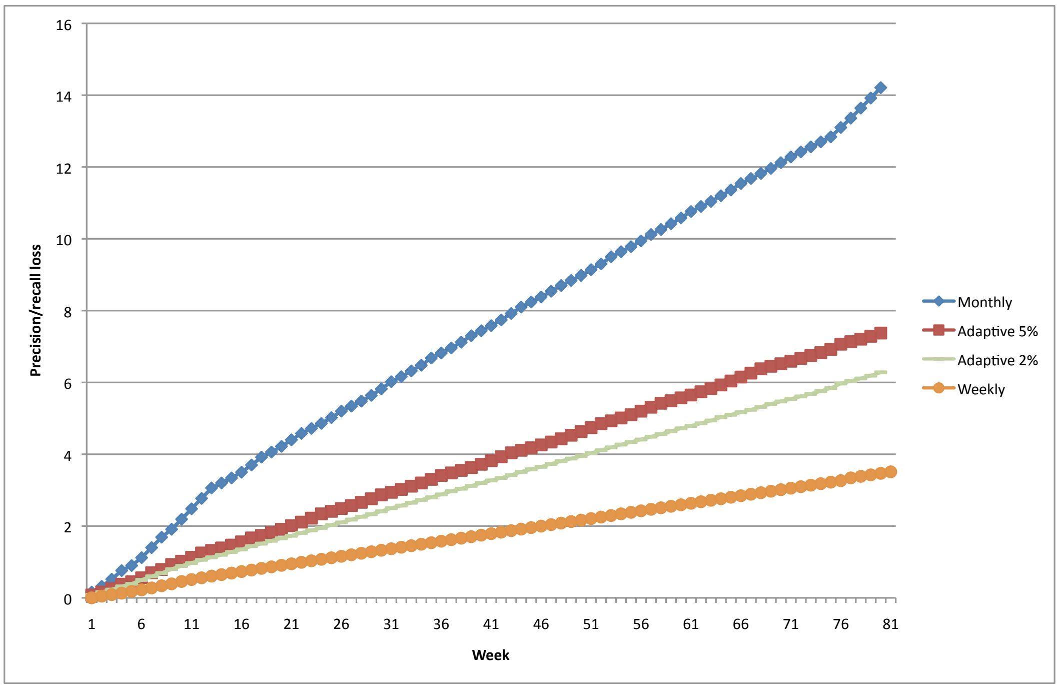

Figure 6.7: Cumulative error on Bibsonomy computed for 75%-25% test set

Chapter 7

Conclusions and Future Work

The main goal of the work presented in this thesis has been the development of a new effective and scalable recommendation algorithm addressing both the *user* and *item cold start problems*. More specifically, we first studied the tri-dimensional relationship between users, items and tags typical of social tagging and we derived a definition of similarity between users and between tags. We then showed how the computed similarities can be used to improve *recommendation efficacy* and reported the results of the extensive evaluation of our proposed Social Ranking (SR) recommendation technique on three different datasets. We analysed how users create new items and showed how preferences of only a small portion of active users (leaders), responsible for the vast majority of the tagged items, can be used to improve the *system scalability.* This further investigation resulted in Clustered Social Ranking (CSR), a scalable recommendation technique capable of effectively suggesting relevant items to users while improving the system *scalability.* Finally, while traditional evaluation methods are based on static data, we proposed a temporal analysis of the performance of CSR. We first investigated the growth of the number of users, items and tags in the system over time and how it influenced CSR performance. We then demonstrated how this information can be used to define an adaptable updating technique to decide whether the benefits of an update of the data structures modelling the system outweigh the corresponding costs.

In this last chapter, we revise the main contributions of this thesis, provide a critical evaluation of the obtained results and discuss some new challenges that will be the target of future developments.

7.1 Contributions

The following is an overview of the main contributions of this thesis.

7.1.1 Cold Start Users and Items

We have provided a solution to the problem of suggesting even unpopular items to both known and new users in the system. By studying the tri-dimensional relationship between users, items and tags typical of social tagging, we derived a definition of similarity between users and between tags which is leveraged when producing recommendations. The relationship between users is based on the set of commonly used tags and can be exploited to find suitable recommenders for the querying user. The relationship between tags is based on the common items they have been associated with and can be exploited to find the best items to recommend.

Our solution addresses the *cold start problems*. First, it exploits tag expansion to find both unpopular and new items. A new item tagged with few tags has an higher probability to be recommended if one of the expanded query tags has been used to tag it. Second, it uses the similarity between the expanded query tags and all tags used by users to rank the recommended items. Finally, it improves the accuracy of the returned results using users' similarities. Note that even if similar users cannot be found for *cold start* querying users, an accurate recommendation list can still be found through tags' similarities. This solution has been implemented into the Social Ranking recommendation algorithm and evaluated on three different social tagging websites, namely CiteULike, Bibsonomy and MovieLens, to prove its efficacy in different scenarios.

7.1.2 Scalability

Based on the observation that the vast majority of items is created by a rather small portion of users (*leaders*), we have designed a scalable technique capable of producing effective recommendations while relying only on a small set of meaningful information. The proposed solution first identifies who the leaders are and clusters them in domains of interest based on their past tagging activity. The opinions of only this small set of selected leaders are then exploited to provide recommendations for both known and new users.

This model has been implemented into the Clustered Social Ranking recommendation algorithm and evaluated on three different social tagging websites, namely CiteULike, Bibsonomy and MovieLens, to prove both its efficacy and scalability in different scenarios.

7.1.3 Adaptive Update

We have devised a technique to decide when the data structures used by the recommendation algorithms must be updated, depending on the growth of the number of users, items and tags over time. Most recommender systems (including SR and CSR) rely on pre-computed data structures that must be kept up-to-date to provide accurate recommendations. We thus provided an analysis of the growth of a tag-based dataset showing the rate at which new users, tags and items appear in the system. We then evaluated the loss of efficacy that follows data growth if the system is not updated. Results showed that systems do not grow uniformly and that data growth is not the same for users, tags and items. We defined a new adaptive technique capable of deciding when the data structures must be updated depending on data growth. This allowed us to perform system updates only when expected benefits outweigh costs.

We have implemented the technique and evaluated it over the Bibsonomy dataset.

7.2 Caveat

Despite the contributions of our work, we must also turn our attention on the possible weaknesses of our proposals so that future efforts can be properly directed. Our evaluation is based on a limited number of datasets and we cannot claim that our conclusions would apply for all existing social tagging websites. However, the algorithms (SR/CSR) we have proposed in this thesis are general and do not depend on the type of items to be recommended. In fact our experiments proved the efficacy of our implementation for three different datasets describing the tagging activity of users on papers, URLs and movies. We can therefore conclude that (C)SR works on all datasets with similar characteristics of tags' distribution and growth as described in Sections 2.2 and 6.1.

7.3 Future Work

Future work will focus on two different threads of research: *model improvement* and *new challenges in tag based environment.*

7.3.1 Model Improvement

Performance improvements might be obtained by modifying the model in some of its aspects:

- **Users and tags similarity measures**: our choice of using cosine-similarity to compare items and users was motivated by the analysis described in [Lathia et al., 2008] and [Sarwar et al., 2001], where the authors state that it gives the best results for recommender systems. However, we could use other measures [Manning et al., 2008] such as the Pearson correlation, the Jaccard similarity and the Euclidean and

Manhattan distances. Using different measures could result in different performance. Extensive experiments will be necessary to understand which solution works best.

- **Clustering techniques**: rather than using the fuzzy C-Means algorithm to cluster users in groups according to their interests, we could use other techniques specifically developed to deal with high-dimensional data. Moreover, while we clustered users according to the items they tagged, the clustering could also be based on the tags they used. Again, extensive experiments will be necessary to understand which technique could give the best improvements.
- **Selection of leaders**: our current implementation selects *leaders* according to the amount of items they tagged. Other approaches may be considered, for example all users belonging to the neighborhood of more then a certain number of other users could be considered *leaders*. Some websites (e.g., Rotten Tomatoes) explicitly decide who the leaders are on the base of their professional activity. Again, the effectiveness of these changes will have to be tested with extensive experiments.

7.3.2 New Challenges in Tag-based Environments

Many new challenges and requirements have been recently raised for tag-based environments that could be interesting to explore:

- **Ontology mapping**: techniques have been proposed to automatically map the tags users associate with items into an ontology of concepts [Passant, 2007] to remove ambiguities between synonyms and polysemys. The ontology could also be useful to find relationships between tags and items and thus to improve the efficacy of the recommendation algorithm. Performance of (C)SR could benefit from an ontology both when expanding queries and when calculating the similarity between tags. For example, the similarity of two tags t_i and t_j could be increased if they both belonged to the same branch in the ontology tree or if they shared a common ancestor. In other words, we can consider the similarity between the semantics of two tags to increase their similarity score.
- **Usage of metadata**: most of the items shared on the Web carry a bag of descriptive metadata. As for ontologies, this metadata can be used to find useful relationships between tags and items and thus to improve the efficacy of the recommendation algorithm. Rather than just using users' and tags' similarities, (C)SR could also calculate items' similarity based on similar metadata. This calculation obviously depends on the type of metadata associated with items. For example a free text description of an item could not be used by any solution, while a structured set of information (e.g., expressed as an XML structured document) could be interpreted by an automatic process.

- **Scalability improvement**: scalability can be improved either parallelizing or distributing the recommendation process (http://mahout.apache.org/). Further optimizations or improvements can be implemented on top of our clustered version of Social Ranking with no interference. A further speed-up in the calculation of recommendations could be of utter importance for very large datasets as can often be found online.

Bibliography

[Agresti, 1984] Agresti, A. (1984). *Analysis of Ordinal Categorical Data*. John Wiley and Sons.

[Aha et al., 1991] Aha, D., Kibler, D., and Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, 6(1):37–66.

[Amatrian et al., 2009] Amatrian, X., Lathia, N., M.Pujol, J., Kwak, H., and Oliver, N. (2009). The wisdom of the few. Proceedings of SIGIR 2009, Boston, Massachusetts.

[Anand et al., 2007] Anand, S., Kearney, P., and Shapcott, M. (2007). Generating semantically enriched user profiles for web personalization. *ACM Trans. Internet Technol.*, 7(4):22+.

[Anand and Mobasher, 2007] Anand, S. and Mobasher, B. (2007). Contextual recommendation. In Berendt, B., Hotho, A., Mladenic, D., and Semeraro, G., editors, *From Web to Social Web: Discovering and Deploying User and Content Profiles*, volume 4737 of *Lecture Notes in Computer Science*, chapter 8, pages 142–160. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Anderson, 2006] Anderson, C. (2006). *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion.

[Andriy et al., 2008] Andriy, S., Jonathan, G., Bamshad, M., and Robin, B. (2008). Personalized recommendation in social tagging systems using hierarchical clustering. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 259–266, New York, NY, USA. ACM.

[Baker, 2008] Baker, L. (2008). Delicious.com relaunches: Enhanced speed, search and design with no dots. http://www.searchenginejournal.com/deliciouscom-relaunches-enhanced-speed-search-design-with-no-dots/7403/.

[Bezdek, 1981] Bezdek, J. (1981). Pattern recognition with fuzzy objective function algorithms (advanced applications in pattern recognition). Springer.

[Bogers and van den Bosh, 2009] Bogers, T. and van den Bosh, A. (2009). Collaborative and content-based filtering for item recommendation on social bookmarking websites. In *RecSys '09: Proceedings of the 2009 ACM conference on Recommender systems*, New York, NY, USA. ACM.

[Breese et al., 1998] Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, San Francisco. Morgan Kaufmann.

[Broder, 1990] Broder, A. (1990). Strategies for efficient incremental nearest neighbor search. *Pattern Recognition*, 23:171–178.

[Capocci and Caldarelli, 2008] Capocci, A. and Caldarelli, G. (2008). Folksonomies and clustering in the collaborative system citeulike. *Journal of Physics A: Mathematical and Theoretical*, 41(22):224016–224023.

[Cattuto et al., 2008] Cattuto, C., Benz, D., Hotho, A., and Stumme, G. (2008). Semantic grounding of tag relatedness in social bookmarking systems. In Sheth, A. P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T. W., and Thirunarayan, K., editors, *The Semantic Web – ISWC 2008*, volume 5318 of *Lecture Notes in Computer Science*, pages 615–631, Berlin/Heidelberg. Springer.

[Cattuto et al., 2007] Cattuto, C., Loreto, V., and Pietronero, L. (2007). Collaborative tagging and semiotic dynamics. *PNAS*, 104(5):1461–1464.

[Collins-Thompson, 2009] Collins-Thompson, K. (2009). Reducing the risk of query expansion via robust constrained optimization. In *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*, pages 837–846, New York, NY, USA. ACM.

[Connor and Herlocker, 2001] Connor, M. and Herlocker, J. (2001). Clustering items for collaborative filtering.

[Dean and Ghemawat, 2004] Dean, J. and Ghemawat, S. (2004). Mapreduce: simplified data processing on large clusters. In *In OSDI'04: Proceedings of the 6th conference on Symposium on Opearting Systems Design &amp; Implementation.*

[Du et al., 2007] Du, N., Wu, B., Pei, X., Wang, B., and Xu, L. (2007). Community detection in large-scale social networks. In *WebKDD/SNA-KDD '07: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 16–25, New York, NY, USA. ACM.

[Emamy and Cameron, 2007] Emamy, K. and Cameron, R. (2007). Citeulike: A researcher's social bookmarking service.

[Faragó et al., 1993] Faragó, A., Linder, T., and Lugosi, G. (1993). Fast nearest neighbor search in dissimilarity spaces. volume 15, pages 957–962. IEEE computer Society.

[Friedman et al., 1975] Friedman, J., Baskett, F., and Shustek, L. (1975). An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, 24:1000–1006.

[Gemmell et al., 2009] Gemmell, J., Schimoler, T., Ramezani, M., and Mobasher, B. (2009). Adapting k-nearest neighbor for tag recommendation in folksonomies. In Anand, S. S., Mobasher, B., Kobsa, A., and Jannach, D., editors, *ITWP*, volume 528 of *CEUR Workshop Proceedings*. CEUR-WS.org.

[Goldberg et al., 1992] Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

[Goldberg et al., 2000] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. (2000). Eigentaste: A constant time collaborative filtering algorithm.

[Golder and Huberman, 2006] Golder, S. A. and Huberman, B. A. (2006). Usage patterns of collaborative tagging systems. *Journal of Information Science*, 32(2):198–208.

[Gunawardana and Shani, 2010] Gunawardana, A. and Shani, G. (2010). Evaluating recommendation systems. In *In RecSys'10: Proceedings of the 4th conference on Recommender Systems*.

[Haase et al., 2004] Haase, P., Ehrig, M., Hotho, A., and Schnizler, B. (2004). Personalized information access in a bibliographic peer-to-peer system. In *In Proceedings of the AAAI Workshop on Semantic Web Personalization*, pages 1–12.

[Halpin et al., 2007] Halpin, H., Robu, V., and Shepherd, H. (2007). The complex dynamics of collaborative tagging. In *Proceedings of the 16th International Conference on World Wide Web*, pages 211–220, New York, NY, USA. ACM Press.

[Hassan-Montero and Herrero-Solana, 2006] Hassan-Montero, Y. and Herrero-Solana, V. (2006). Improving tag-clouds as visual information retrieval interfaces. In *InScit2006: International Conference on Multidisciplinary Information Sciences and Technologies*.

[Herlocker et al., 1999] Herlocker, J., Konstan, J., Borchers, A., and Riedl, J. (1999). An algorithmic framework for performing collaborative filtering. In *SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, NY, USA. ACM.

[Heymann and Garcia-Molina, 2006] Heymann, P. and Garcia-Molina, H. (2006). Collaborative creation of communal hierarchical taxonomies in social tagging systems. Technical Report 2006-10, Stanford University.

[Heymann et al., 2008] Heymann, P., Koutrika, G., and Garcia-Molina, H. (2008). Can social bookmarking improve web search? In *WSDM '08: Proceedings of the international conference on Web search and web data mining*, pages 195–206, New York, NY, USA. ACM.

[Hill et al., 1995] Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

[Hotho et al., 2006] Hotho, A., Jschke, R., Schmitz, C., and Stumme, G. (2006). Information retrieval in folksonomies: Search and ranking. In Sure, Y. and Domingue, J., editors, *The Semantic Web: Research and Applications*, volume 4011 of *LNAI*, pages 411–426, Heidelberg. Springer.

[Jain et al., 1999] Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323.

[Ji et al., 2007] Ji, A.-T., Yeon, C., Kim, H.-N., and Jo, G.-S. (2007). Collaborative tagging in recommender systems. In *AI 2007: Advances in Artificial Intelligence*, pages 377–386, New York, NY, USA. ACM.

[Jschke et al., 2008] Jschke, R., Marinho, L., Hotho, A., Schmidt-Thieme, L., and Stumme, G. (2008). Tag recommendations in social bookmarking systems. *AI Communications*, 21(4):231–247.

[Kaser and Lemire, 2007] Kaser, O. and Lemire, D. (2007). Tag-cloud drawing: Algorithms for cloud visualization. *CoRR*, abs/cs/0703109. informal publication.

[Konstan et al., 1997] Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., and Riedl, J. (1997). Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87.

[Krulwich, 1997] Krulwich, B. (1997). Lifestyle finder: Intelligent user profiling using large-scale demographic data. *AI Magazine*, 18(2):37–45.

[Kuchinskas, 2005] Kuchinskas, S. (2005). Flickr to add print to photo service. http://www.internetnews.com/ec-news/article.php/3512866/Flickr-to-Add-Print-to-Photo-Service.htm.

[Lam et al., 2008] Lam, X. N., Vu, T., Le, T. D., and Duong, A. D. (2008). Addressing cold-start problem in recommendation systems. In *ICUIMC '08: Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 208–211, New York, NY, USA. ACM.

[Lathia et al., 2008] Lathia, N., Hailes, S., and Capra, L. (2008). The effect of correlation coefficients on communities of recommenders. In *Proceedings of 23rd Annual ACM Symposium on Applied Computing.*

[Leung et al., 2007] Leung, C. W.-k., Chan, S. C.-f., and Chung, F.-l. (2007). Applying cross-level association rule mining to cold-start recommendations. In *WI-IATW '07: Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, pages 133–136, Washington, DC, USA. IEEE Computer Society.

[Linden et al., 2003] Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80.

[Manning et al., 2008] Manning, C., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval.* Cambridge University Press.

[Mardia et al., 1979] Mardia, K., Kent, J., and Bibby, J. (1979). Multivariate analysis. Academic Press.

[Massa and Avesani, 2007] Massa, P. and Avesani, P. (2007). Trust-aware recommender systems. In *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 17–24, New York, NY, USA. ACM.

[Mobasher et al., 2001] Mobasher, B., Dai, H., Luo, T., and Nakagawa, M. (2001). Effective personalization based on association rule discovery from web usage data. In *WIDM '01: Proceedings of the 3rd international workshop on Web information and data management*, pages 9–15, New York, NY, USA. ACM.

[Mull, 2006] Mull, M. (2006). Characteristics of high-volume recommender systems. In *Recommenders Workshop*, Bilbao, Spain.

[Nakamoto et al., 2007] Nakamoto, R., Nakajima, S., Miyazaki, J., and Uemura, S. (2007). Tag-based contextual collaborative filtering. In *18th IEICE Data Engineering Workshop.*

[Omran et al., 2007] Omran, M. G., Engelbrecht, A. P., and Salman, A. A. (2007). An overview of clustering methods. *Intell. Data Anal.*, 11(6):583–605.

[Pan et al., 2009] Pan, J., Taylor, S., and Thomas, E. (2009). Reducing ambiguity in tagging systems with folksonomy search expansion. In *6th Annual European Semantic Web Conference (ESWC2009)*, pages 669–683.

[Passant, 2007] Passant, A. (2007). Using ontologies to strengthen folksonomies and enrich information retrieval in weblogs. In *Proceedings of International Conference on Weblogs and Social Media.*

[Peng et al., 2004] Peng, H., Bo, X., Fan, Y., and Ruimin, S. (2004). A scalable p2p recommender system based on distributed collaborative filtering. *Expert systems with applications.*

[Polat and Du, 2003] Polat, H. and Du, W. (2003). Privacy-preserving collaborative filtering using randomized perturbation techniques. In *The Third IEEE International Conference on Data Mining (ICDM'03)*, Melbourne, FL.

[Rajeev et al., 1999] Rajeev, S. G., Rastogi, R., and Shim, K. (1999). Rock: A robust clustering algorithm for categorical attributes. In *Information Systems*, pages 512–521.

[Rendle et al., 2009] Rendle, S., Marinho, L. B., Nanopoulos, A., and Schmidt-Thieme, L. (2009). Learning optimal ranking with tensor factorization for tag recommendation. In *The 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09)*, pages 727–736, New York, NY, USA. ACM.

[Resnick et al., 1994] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: an open architecture for collaborative filtering of netnews. In *CSCW '94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, New York, NY, USA. ACM.

[Ruan and Zhang, 2008] Ruan, J. and Zhang, W. (2008). Identifying network communities with a high resolution. *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)*, 77(1).

[Sarwar et al., 2001] Sarwar, B., Karypis, G., Konstan, J., and Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA. ACM.

[Sarwar et al., 2000] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Application of dimensionality reduction in recommender systems–a case study. In *ACM WebKDD Workshop.*

[Sarwar et al., 2002] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2002). Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the Fifth International Conference on Computer and Information Technology.*

[Schaeffer, 2007] Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, 1(1):27 – 64.

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

[Seco et al., 2004] Seco, N., Veale, T., and Hayes, J. (2004). An intrinsic information content metric for semantic similarity in wordnet. In *ECAI '04: 16th European Conference on Artificial Intelligence.*

[Sen et al., 2006] Sen, S., Lam, S. K., Rashid, A. M., Cosley, D., Frankowski, D., Osterhouse, J., Harper, M. F., and Riedl, J. (2006). Tagging, communities, vocabulary, evolution. In *Proceedings of the 20th Conference on Computer Supported Cooperative Work*, pages 181–190, New York, NY, USA. ACM Press.

[Shardanand and Maes, 1995] Shardanand, U. and Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217.

[Shen and Wu, 2005] Shen, K. and Wu, L. (2005). Folksonomy as a complex network.

[Sheung-On and Lui, 2006] Sheung-On, C. and Lui, A. K. (2006). Web information retrieval in collaborative tagging systems. pages 352–355.

[Su and Khoshgoftaar, 2009] Su, X. and Khoshgoftaar, T. (2009). A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:2.

[Sudipto et al., 1998] Sudipto, G., Rajeev, R., and Kyuseok, S. (1998). Cure: an efficient clustering algorithm for large databases. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 73–84, New York, NY, USA. ACM.

[Suryavanshi et al., 2005] Suryavanshi, B. S., Shiri, N., and Mudur, S. (2005). A fuzzy hybrid collaborative filtering technique for web personalization. In *ITWP '05: Proceedings of the 3rd Workshop on Intelligent Techniques for Web Personalization*, New York, NY, USA. ACM.

[Symeonidis et al., 2008] Symeonidis, P., Nanopoulos, A., and Manolopoulos, Y. (2008). Tag recommendations based on tensor dimensionality reduction. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 43–50, New York, NY, USA. ACM.

[Takacs et al., 2009] Takacs, G., Pilaszy, I., and Nemeth, B. (2009). Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research*, (10):623–656.

[Tibshirani et al., 2000] Tibshirani, R., Walther, G., and Hastie, T. (2000). Estimating the number of clusters in a dataset via the gap statistic.

[Tso-Sutter et al., 2008] Tso-Sutter, K. H. L., Marinho, L. B., and Schmidt-Thieme, L. (2008). Tag-aware recommender systems by fusion of collaborative filtering algorithms. In *Proceedings of 23rd Annual ACM Symposium on Applied Computing*, pages 16–20. ACM Press.

[Ungar and Foster, 1998] Ungar, L. and Foster, D. (1998). Clustering methods for collaborative filtering. In *Proceedings of the Workshop on Recommendation Systems*. AAAI Press, Menlo Park California.

[Wang et al., 2006] Wang, J., de Vries, A., and Reinders, M. (2006). Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, New York, NY, USA. ACM.

[Wetzker et al., 2009] Wetzker, R., Umbrath, W., and Said, A. (2009). A hybrid approach to item recommendation in folksonomies. In *ESAIR '09: Proceedings of the WSDM '09 Workshop on Exploiting Semantic Annotations in Information Retrieval*, pages 25–29, New York, NY, USA. ACM.

[Xue et al., 2005] Xue, G. R., Lin, C., Yang, Q., Xi, W., J.Zeng, H., Yu, Y., and Chen, Z. (2005). Scalable collaborative filtering using cluster-based smoothing. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121, New York, NY, USA. ACM.

[Yang et al., 2002] Yang, Y., Guan, X., and You, J. (2002). Clope: A fast and effective clustering algorithm for transactional data. In *In: Proc of KDD02*, pages 682–687.

[Yeung et al., 2007] Yeung, C. M. A., Gibbins, N., and Shadbolt, N. (2007). Mutual contextualization in tripartite graphs of folksonomies. In *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea*, volume 4825 of *LNCS*, pages 960–964, Berlin, Heidelberg. Springer Verlag.

[Zanardi and Capra, 2008] Zanardi, V. and Capra, L. (2008). Social ranking: uncovering relevant content using tag-based recommender systems. In *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*, pages 51–58, New York, NY, USA. ACM.

[Zhang and Srihari, 2004] Zhang, B. and Srihari, S. (2004). Fast k-nearest neighbor classification using cluster-based trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:525–528.

[Zhang et al., 1996] Zhang, T., Ramakrishnan, R., and Livny, M. (1996). Birch: An efficient data clustering method for very large databases.