Using Data Mining for Facilitating User Contributions in the Social Semantic Web

Zur Erlangung des akademischen Grades eines

Doktors der Wirtschaftswissenschaften
(Dr. rer. pol.)
von der Fakultät für
Wirtschaftswissenschaften
Karlsruher Institute für Technologie
genehmigte
DISSERTATION
von
Maryam Ramezani

Tag der mündlichen Prüfung: 25 Feb 2011
Referent: Prof. Dr. Rudi Studer
Korreferent: Prof. Dr. Alexander Maedche
2011 Karlsruhe

Abstract

Social Web applications have emerged as powerful applications for Internet users allowing them to freely contribute to the Web content, organize and share information, and utilize the collective knowledge of others for discovering new topics, resources and new friends. While social Web applications such as social tagging systems have many benefits, they also present several challenges due to their open and adaptive nature. The amount of user generated data can be extremely large and since there is not any controlled vocabulary or hierarchy, it can be very difficult for users to find the information that is of their interest. In addition, attackers may attempt to distort the system's adaptive behavior by inserting erroneous or misleading annotations, thus altering the way in which information is presented to legitimate users. This thesis utilizes data mining and machine learning techniques to address these problems. In particular, we design and develop recommender systems to aid the user in contributing to the Social Semantic Web. In addition, we study intelligent techniques to combat attacks against social tagging systems. In our work, we first propose a framework that maps domain properties to recommendation technologies. This framework provides a systematic approach to find the appropriate recommendation technology for addressing a given problem in a specific domain. Second, we improve existing graph-based approaches for personalized tag recommendation in folksonomies. Third, we develop machine learning algorithms for recommendation of semantic relations to support continuous ontology development in a social semantic Web environment. Finally, we introduce a framework to analyze different types of potential attacks against social tagging systems and evaluate their impact on those systems.

Zusammenfassung

Social-Web-Applikationen haben sich als leistungsstarke Anwendungen erwiesen, die es Internet-Nutzern ermöglichen, flexibel neue Web-Inhalte bereitzustellen, Informationen zu organisieren und auszutauschen, und das kollektive Wissen anderer für die Entdeckung neuer Themen und Ressourcen und das Finden neuer Freunde zu nutzen. Während Social-Web-Anwendungen, wie zum Beispiel Social-Tagging-Systeme, viele Vorteile haben, beinhalten sie auch einige Herausforderungen aufgrund ihrer offenen und anpassungsfähigen Natur. Die Menge an nutzergenerierten Daten kann extrem groß sein und da es kein kontrolliertes Vokabular oder Hierarchien gibt, kann es für die Nutzer sehr schwierig sein, die Informationen zu finden, die von Interesse sind. Darüber hinaus können Angreifer versuchen das adaptive Verhalten des Systems zu verzerren, indem Sie falsche oder irreführende Annotationen verwenden, um damit die Art und Weise zu modifizieren, wie die Informationen für den rechtmäßigen Benutzer präsentiert werden. Diese Arbeit nutzt Data Mining und Techniken des maschinellen Lernens, um diese Probleme anzugehen. Insbesondere entwerfen und entwickeln wir Recommender-Systeme, um Nutzern eine Hilfe zu geben ihren Beitrag zum Social Web leisten zu können. Zusätzlich untersuchen wir intelligente Techniken, um Angriffe gegen Social-Tagging-Systeme zu bekämpfen. In unserer Arbeit schlagen wir zuerst ein Framework vor, welches Problemstellungstypen auf Typen von Recommendation-Technologien abbildet. Dieses Framework bietet einen systematischen Ansatz, um die adäquate Recommendation-Technologie für die Bewältigung eines bestimmten Problems in einer spezifischen Domäne zu finden. Zweitens verbessern wir bestehende graph-basierte Ansätze für personalisierte Tag-Empfehlungen in Folksonomies. Drittens entwickeln wir Machine-Learning-Algorithmen für die Empfehlung von semantischen Relationen, die der kontinuierlichen Entwicklung von Ontologien in einer Social-Semantic-Web-Umgebung dienen. Schließlich führen wir ein Framework ein, um verschiedenen Arten von möglichen Angriffen auf Social-Tagging-Systemen zu analysieren und bewerten ihre Auswirkungen auf diese Systeme.

Acknowledgments

First, I would like to thank my advisor Prof. Dr. Rudi Studer for his advise and support to improve my work. Being a member of prof.Studer's group, I had the opportunity to work with an amazing supportive team. Among all, I would like to give my deepest appreciation to Dr. Valentin Zacharias who supervised my work, helped me to improve it and guided me in all the steps of my PhD at Karlsruhe Institute of Technology. This thesis would not have been possible without him and his support.

This thesis is a combination of my research efforts at different research institutes that I have visited. I had the chance to work with many wonderful professors and researchers. I learned a lot from Prof. Bamshad Mobasher and Professor Robin Burke while I was at center for Web intelligence at DePaul University, Chicago, IL. They both supported me to have an excellent research opportunity while teaching there. Working with them was a great experience that taught me scientific thinking and the steps to do valuable research. I worked together with an amazing, creative team including Jonathan Gemmell, Thomas Schimoler, J.J. Sandvig, and Runa Bhaumik. I learned a lot from all of them and had a great time in our meetings and scientific discussions. Thanks to all of them for this great experience.

I spent a great summer at IBM Watson Research Center, Hawthorne, NY working on recommender systems and I would like to thank Lawrence Bergman and Rich Thompson for their mentorship and support during my time there. I spent some months at FZI (Forschungszentrum Informatik), Karlsruhe where I contributed to Soboleo social bookmarking system. Thanks to Simone Braun for supporting me at those times. Last but not the least, I spent more than a year at SAP research while working on my thesis and I would like to thank my mentors and managers who supported me there. Many thanks to Dr. Hans Friedrich Witschel, Dr. Harald Vogt, and Dr. Zoltan Nochta for their support during my PhD time at SAP research. Special thanks go to Dr. Hans Friedrich Witschel for his willingness to go through my thesis carefully and for giving me valuable feedback.

I would like to thank the members of my committee for taking the time out of their busy schedules to review my work and attending my defense: Prof. Dr. Alexander Maedche, Prof. Dr. Hartmut Schmeck, and Prof. Dr. Andreas Geyer-Schulz.

Finally, I would like to passionately thank the most important people of my life: my great family who has been the greatest support in all steps of my life. Thanks to my wonderful husband who was my biggest emotional support at each and every second of my PhD work. Thanks to my wonderful parents who taught and encouraged me to be an explorer, to have courage to learn new things and to be confident of myself that I can do it.

Contents

Chapter 1

Introduction

Social Web applications have emerged as powerful applications for Internet users allowing them to freely contribute to the Web content, to organize and to share information, and to utilize the collective knowledge of others for discovering new topics, resources and new friends.

In social Web applications, information access functions such as search, navigation, and resource sharing are usually supported by annotations, labels that users apply to resources. These labels can be numerical ratings such as those used in many standard recommender systems; simple meta-data in the form of short tags such as those supported by Delicious[1], Flickr[2], and many other social tagging systems; or they might be more structred in the form of categories in Wikipedia[3]. Many of these applications support their users by providing recommendations.

Recommender systems reduce the burden of navigating large information spaces and aid users in discovering new items of their interest. However, single-function recommender systems, which connect users with relevant resources are being progressively replaced by more complex and dynamic applications in which social Web and social networking play an increasingly important role.

This thesis aims to use machine learning and data mining algorithms, particularly, recommender systems to assist users of social Web applications to bring meaningful contributions to the system with minimum effort. In addition, we want to understand the conditions under which information access in social Web applications can be expected to produce useful results, and in particular, how these functions can be insulated from adversaries, even in systems that are open to the public.

In this chapter, we first present our motivations and research questions in section 1.1.

[1]www.delicious.com

[2]www.flickr.com

[3]www.wikipedia.org/

Next, we present our contributions and the thesis overview in section 1.2.

1.1 Motivations and Research Questions

In this thesis, we investigate how machine learning and data mining algorithms can be applied to social Web applications in order to improve their usability. For this purpose, we specifically investigate the role of recommender systems in social Web applications, particularly in social tagging systems. In this section, we outline the objectives and the research questions that this thesis aims to address.

Objective 1: In the first step of this thesis, we study the problem of matching recommender systems technologies to different domains. With the rapid development of recommender system technologies in the recent years, it has become difficult for developers to determine which technology is suitable to a particular context. To alleviate this difficulty, we propose a framework that organizes the space of recommendation problems and provides a systematic approach to finding the appropriate recommendation technology for addressing a given problem in a specific domain. The research questions we try to answer in this part of the thesis are the following.

- What are the main characteristics of a domain that influence the choice of selecting the appropriate recommendation technology?
- What are the main knowledge sources in recommendation systems and how do they relate to the recommendation technology?
- How can we map the domain characteristics to recommendation technologies?

Objective 2: Once we have a good understanding of recommender system technologies and their applications, we focus our attention on building recommenders in the social tagging domain. Social tagging applications allow users to annotate online resources, resulting in a complex three dimensional network of interrelated users, resources and tags often called a Folksonomy. In order to develop a recommender application for such a system, the first step is to create a model of the folksonomy that takes into account the information flow between users, resources and tags. We suggest a directed graph to model the folksonomy and apply an adaptation of PageRank algorithm on the graph for tag recommendation. Our goal in this part of the thesis is to improve the link analysis in folksonomies for tag recommendation and the research question we try to address are the following.

- How can we model the folksonomy as a graph so that we can capture the flow of information?

- How can we use the model for tag recommendation?
- Does the proposed model produce better recommendation than the previous approaches?

Objective 3: Bridging the gap between folksonomies and ontologies as a research problem has recently received attention by many researchers in the semantic community. Applying semantic Web technologies to the data of the social Web can help users in search and navigation. On the other hand, the collaborative environment of social tagging systems can also provide a suitable platform for developing ontologies. In this thesis, we investigate how recommender systems can help users to collaboratively develop an ontology. We propose a machine learning algorithm that utilizes a seed concept hierarchy to automatically learn from the existing relations among concepts. The algorithm recommends semantic relations between a new concept (entered by users) and the existing concepts in the hierarchy. The research questions we try to answer are the following.

- How can we aggregate user activities in a social tagging environment to infer semantic similarity?
- How can we use machine learning to learn from the existing semantic relations in a concept hierarchy to predict semantic relations between a new concept and existing concepts?
- How can the algorithm support collaborative ontology development?

Objective 4: Although flexibility and openness of social tagging systems has attracted millions of users, these properties also introduce challenges to the system. One of the major challenges that has received little attention from research community is the security of such systems. Attackers may attempt to distort the systems adaptive behavior by inserting erroneous or misleading annotations, thus altering the way in which information is presented to legitimate users. In this thesis, we systematically study this issue. We are interested to answer the following questions.

- How can we systematically study the problem of attacks against social tagging systems?
- How can we evaluate the impact of attacks in social tagging systems?
- What model of attacks are more successful?
- How many malicious users can a tagging system tolerate before results significantly degrade?
- How much effort and knowledge is needed by an attacker to attack the system?

1.2 Thesis overview and contributions

In this section, we present an overview of the thesis and outline our contributions. The following subsections provide a brief description of each of the main chapters of the thesis.

1.2.1 Matching Recommendation Technology and Domains

The goal of this chapter is to assist researchers and developers to identify the recommendation technology that is applicable to different domains of recommendation. This chapter is a conceptual contribution to the field of recommender systems and it provides a comprehensive understanding of how recommender systems can be applied in different domains. In this chapter, we adopt a taxonomy of knowledge sources in recommendation from existing literature and match those knowledge sources to recommendation technologies. We identify the domain characteristics that influence on selection of recommendation technologies and map those characteristics to recommendation technologies based on the knowledge sources they require.

Contributions

- Extensive survey on the recommendation literature
- Identification of domain characteristics that influence on selecting and applying recommendation technology
- Mapping the domain characteristic to recommendation technologies

Impact

- This work has provided the basic fundamentals for development of recommender systems in IBM T.J Watson Research Center.
- Preliminary results of the work have been published as a paper in the workshop on recommendation and collaboration at the Intelligent User Interfaces Conference 2008 [167].
- An extended version of the work has been published in the Recommender Systems Handbook providing a guide for research scientists and practitioners in the recommender system area [39].

1.2.2 Improving Link Analysis for Tag Recommendation in Social Tagging Systems

Tag recommendation is one of the techniques that can help reduce noise, redundancy, and ambiguity in social tagging systems. Many researchers have developed different techniques for tag recommendation. Our goal in this chapter is to improve existing tag recommendation techniques by introducing a new model of the folksonomy graph.

Contributions

- Developed an approach to model a folksonomy as a weighted directed graph.
- Used the proposed directed graph and an adaptation of PageRank for tag recommendation.
- Evaluated the proposed algorithm with data set from popular tagging applications such as Bibsonomy, Citeulike and Delicious and showed improvement over popular existing graph-based algorithms.

Impact

- Improvement on existing tag recommendation algorithms
- The results of this work is published in the workshop on recommender systems and the social Web [169] and as a rising scholar paper in the Journal of Emerging Technologies in Web Intelligence [166].

1.2.3 Using Recommender Systems for Continuous Ontology development

In this chapter we develop a recommender system to support continuous ontology development. Our goal is to develop a recommendation algorithm in a Web 2.0 platform that supports end users in the collaborative development of an ontology. The developed algorithm uses an existing seed hierarchy, the concepts already present and the sub/super concept relations between them to place a given new concept in the hierarchy. Thus, the output of the algorithm consists of potential super-concepts for a new concept.

Contributions

- Developed a machine learning algorithm that can learn from the existing relations of a concept hierarchy and suggest the potential super concepts for a new concept.
- Experimental results from the proposed algorithm with data sets from Wikipedia show excellent results which confirm that the algorithm can be directly used in Wikis to support users for creating sub-super category relations.
- Results from expert evaluations show that the algorithm can suggest accurate high quality semantic relations.

Impact

- The developed algorithm has been implemented in the social semantic bookmarking system SOBOLEO [229] and is being used by users.
- The algorithm has been partly implemented into SAP Floyd system. Floyd is a case management system used by market researchers. Users receive recommendations on the potential semantic relations for a new term.
- The algorithm can be directly used in Wikis to support users for creating sub-super category relations.
- The results of this work has been published in the proceedings of 5th IEEE International Conference on Intelligent Systems [173] and 17th International Conference on Knowledge Engineering and Knowledge Management, EKAW 2010 [168].

1.2.4 Combating Attacks Against Social Tagging Environments

In this chapter we discuss the problem of security and robustness in social tagging systems. We introduce a framework to model the navigation channels in social tagging systems and we identify different types of potential attacks against the system through different navigation channels. We propose approaches to evaluate the impact of attacks. We model different attack types and experiment their impact using dataset from popular social tagging systems. This chapter provides a strong conceptual understanding of the possible attacks against social tagging systems and the experimental section of the chapter helps identify the types of attacks that are more successful in changing the system behavior. Gaining a fundamental understanding of the nature and impact of such attacks can lead to more secure and robust social Web applications.

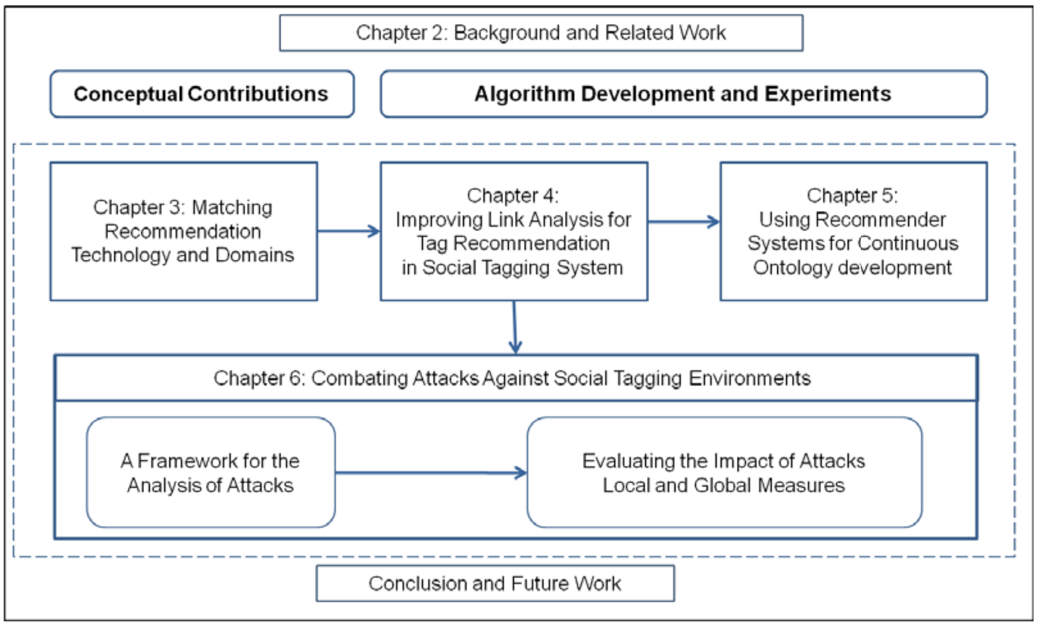

Chapter 2: Background and Related Work
Conceptual Contributions
Algorithm Development and Experiments
Chapter 3: Matching Recommendation Technology and Domains
Chapter 4: Improving Link Analysis for Tag Recommendation in Social Tagging System
Chapter 5: Using Recommender Systems for Continuous Ontology development
Chapter 6: Combating Attacks Against Social Tagging Environments
A Framework for the Analysis of Attacks
Evaluating the Impact of Attacks Local and Global Measures
Conclusion and Future Work

Figure 1.1: Structure of the thesis

Contributions

- Outlined a framework to model the attacks based on various navigation channels and target elements.
- Introduced evaluation metrics to measure the local and global impact of attacks.
- Implemented and evaluated different type of attacks on data sets from real popular folksonomies such as Delicious and Bibsonomy.

Impact

- This work creates the awareness of the security and manipulation problem in open adaptive systems specifically social tagging systems.
- The results of this work can help social tagging Websites discover the vulnerabilities of their system, inform them about the parts of the system that need more monitoring, guide researchers and developers to develop more robust systems, and provide them with clues for attack detection.

- The results of this work have been published in several workshops including [170, 186] as well as IEEE International Conference on Social Computing 2009 [171].

We start this thesis with a background chapter. We present basic concepts and preliminaries, introduce the notation and briefly survey the related work. Next, in chapter 3 we investigate the problem of matching recommendation technologies and domains. In chapter 4, we present our algorithm to improve graph-based tag recommendation. Chapter 5 presents our approach to assist users for continuous ontology development. Chapter 6 is dedicated to our studies on the problem of attacks against social tagging systems. Finally, chapter 7 concludes the thesis with a summary and an outlook to future possibilities to extend this work. Figure 1.1 shows the structure of the thesis.

Chapter 2

Background

In this chapter we review the basic concepts and terminology used in this thesis. This chapter serves as an introduction into general related work covering the subareas of our work.

We start this chapter in section 2.1 by introducing recommender systems: first, a brief history of the field will be given, followed by description of popular recommendation algorithms. Next, in section 2.3 we introduce social tagging systems and our terminology to model such systems. We survey the current areas of research in social tagging systems including different recommendation approaches. Finally, we discuss several challenges in social tagging systems: ambiguity and redundancy, and attacks against social tagging systems. These challenges are the motivation of our work in the next chapters of this thesis.

2.1 Recommender Systems

The growth of the World Wide Web in the 1990s resulted in an explosive growth of the amount of information available online, outgrowing the ability of individual users to process all this information. This was a motivation for the increase of interest in research fields such as information retrieval (IR) and information filtering (IF).

Information Retrieval originated as a research field in the 1950s focusing on automatically matching user's need (specially in form of a query) against a set of documents. Web search engines are the most visible IR applications today.

Information filtering systems emerged in the 1990s to help users with the information overload problem. These systems generally use long term user profiles to filter out irrelevant or redundant data items and just present the part of information that the user is interested in. Typically, such systems construct a model of users' interests and match that against a stream of information objects. While IR and IF are considered separate research fields, they share many characteristics, such as a focus on analyzing textual content [22].

Recommender systems can be considered as active information filtering systems that attempt to present the user with information items he or she is interested in. Thus, they actively add personalized information items to the information flowing towards the users. Recommender systems were originally defined in [176] as "systems in which people provide recommendations as inputs, which the system then aggregates and directs to appropriate recipients". However, this definition only captures "collaborative filtering" which is a specific algorithm for recommending. The term "recommender systems" evolved to replace and broaden the use of the term collaborative filtering and is defined by Schafer et.al. [190] as systems that specifically recommend lists of products and help users evaluate products. In this definition recommenders are systems that are used in E-commerce sites to suggest products to their customers and to provide consumers with information to help them decide which products to purchase. Schafer et.al. [190] present recommender systems as a way to achieve mass customization in e-commerce. From this perspective, recommender systems can be regarded as specialized data mining systems that have been optimized for interaction with consumers rather than marketers.

As the above definitions suggest, recommender systems are generally applied to e-commerce domains such as movies, music, books, news, images, and web pages. In this thesis, specifically in chapter 3, our perspective on recommender systems is broader than only used in e-commerce applications. We view recommenders as systems that produce individualized recommendations as output or have the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options [32]. In the following section, we present the common recommendation algorithms.

2.2 Recommendation Algorithms

Over the past two decades many different recommendation algorithms have been proposed. Generally, in recommendation literature, all different approaches can be categorized to three main approaches: Collaborative Filtering (CF), Content-based (CB) and knowledge-based (KB) algorithms. In this section, we describe the basics of each recommendation technique and discuss the advantages and disadvantages of each approach.

2.2.1 Collaborative Filtering

Collaborative Filtering (CF) is the most popular and, to date, the most successful recommendation technique in e-commerce applications. CF algorithms usually employ statistical techniques to find like-minded people, known as neighbors, that have a history of agreeing with the target user. Once a neighborhood of users is formed, different algorithms are used to combine the preferences of neighbors to produce a prediction of ratings for not-rated items or a list of top n items as recommendation for the target user. The underlying assumption of the CF approach is that those who agreed in the past tend to agree again in the future.

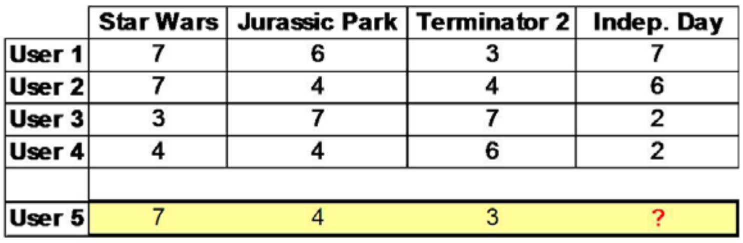
The term “collaborative filtering” was first introduced by Goldberg et al. [73], to describe their Tapestry filtering system in which people collaborate to help each other perform filtering by recording their reactions to documents they read. The reactions, called annotations, can be accessed by other people’s filters. Users had to identify the people who have similar taste to them by themselves. Subsequent CF approaches automated this process of locating the nearest neighbors of the active user. CF algorithm as it is known today, which tries to predict ratings for not rated items based on ratings of nearest neighbors, was presented by Resnick et al. [175] for recommending Usenet articles, and by Shardanand and Maes [192] in their Ringo music recommender system. These were the first to find the distance between users by correlating their rating history in order to (1) determine the most similar neighbors and (2) use those similarities to predict interest in new items.

Since then, many researchers have worked on expanding and improving collaborative filtering systems. Breese et al. [27] introduced the notion of memory-based and model-based algorithms to improve the efficiency of CF techniques. Herlocker et al. [88] performed a large-scale evaluation of collaborative filtering algorithms using different weighting schemes. In the following subsections we explain the basics of collaborative filtering algorithm and describe the differences between user-based, item-based, memory-based and model-based algorithms.

Overview of the Collaborative Filtering Process

The task of collaborative filtering algorithms is to predict how well a user will like an item that he has not rated given his historical preferences together with the historical preferences of a community of users. User preferences can be explicit statements provided intentionally by the user such as ratings or implicitly inferred from user behavior such as browsing the web pages. The problem space can be formulated as a $n \times m$ matrix, R, for n users versus m items where cell R_{ij} of the matrix represents the rating of user i on item j. Under this formulation, the problem is to predict values for empty cells. In general, this matrix is very sparse since each user usually has rated a small percentage of total number of items. Figure 2.1 shows a simplified example of user-item matrix for movie ratings. In this example, the recommender systems attempts to provide a prediction for Indep.Day for user 5.

The **user-based** collaborative filtering, tries to predict the rating of an item by looking at the “nearest neighbors” of the target user. In this example, based on the comparison of rating history of user 5 with other users, user 2 has the most similar rating history to user 5. They have closely agreed on all the movies that they have both seen. As a result, user 2’s opinion



	Star Wars	Jurassic Park	Terminator 2	Indep. Day
User 1	7	6	3	7
User 2	7	4	4	6
User 3	3	7	7	2
User 4	4	4	6	2
User 5	7	4	3	?

Figure 2.1: A simplified example of user-item matrix for movie ratings

about Indep. Day will influence the prediction for user 5. For simplification, in this example, if the system decides only based on one nearest neighbor, then the prediction for user 5 will be equal to 6 for Indep.Day.

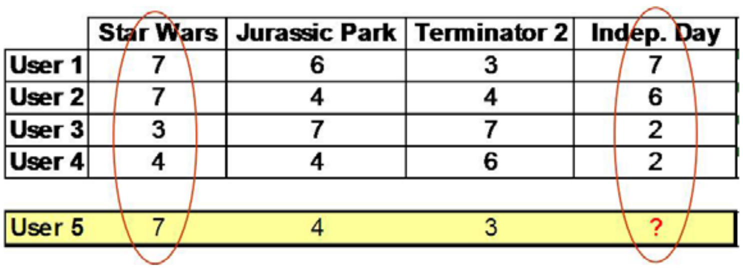
Herlocker et al. [88] separates the user-based method into three steps: (1) Weight all users with respect to similarity with the active user. The most common technique to find similarity among users is Pearson correlation. (2) Select a subset of users to use as the nearest neighbors for prediction. (3) Normalize ratings and compute a prediction using a weighted combination of selected nearest neighbors.

Although user-based collaborative filtering has been very successful, it has some potential challenges including sparsity and scalability. In practice, in many commercial systems, even active users may have purchased or rated under 1% of the items. Thus, a user-based recommender system can hardly find nearest neighbors for a particular user and thus it will be unable to make any item recommendations. To solve this problem, Sarwar etal. [188] proposed the **item-based** collaborative filtering.

Item-based collaborative filtering looks at the set of items the target user has rated and computes how similar they are to the target item i and then selects k most similar items. Once the most similar items are found, the prediction is then computed by taking a weighted average of the target user's ratings on these similar items. Figure 2.2 shows the same user-item matrix as in the previous example. Based on user-ratings the Indep.Day is most similar to Star Wars. Thus, the system predicts that user 5 will have a similar opinion about these movies and predicts 7 for Indep.Day (if we consider only one similar item). In practice, the prediction is based on weighted average of the k most similar items. The recommender system used in Amazon.com (people who bought this, also bought this) works on the same principle.

Memory-based vs. model-based collaborative filtering

CF algorithms are commonly divided into two types, memory-based and model-based algorithms. Memory-based algorithms are similar to lazy machine learning algorithms that



	Star Wars	Jurassic Park	Terminator 2	Indep. Day
User 1	7	6	3	7
User 2	7	4	4	6
User 3	3	7	7	2
User 4	4	4	6	2
User 5	7	4	3	?

Figure 2.2: Item-based collaborative filtering finds similar items based on users rating history

postpone the task of building a model to the time of recommendation generation. At generation time, they scan the entire database to produce a prediction. Generally, user-based CF is considered as memory-based since all the computations are done real time at the time of user interaction.

Model-based CF algorithms learn an aggregated model of user behavior in advance and use this model to generate recommendations. The model building process can be performed by different machine learning algorithms including Bayesian learning, clustering, association rule mining, and factor models.

Bayesian networks create a model based on a training set with a decision tree where a node corresponds to each item. The states of each node correspond to the possible vote values for each item. The model can be built off-line. The resulting model is small, fast, and essentially as accurate as nearest neighbor methods [27]. Bayesian networks may prove practical for environments in which knowledge of user preferences changes slowly with respect to the time needed to build the model but are not suitable for environments in which consumer preference models must be updated rapidly or frequently [190] since a new model has to be built every time the preferences change.

Clustering techniques work by identifying groups of users who have similar preferences. Once the clusters are created, predictions for an individual can be made by averaging the opinions of the other users in that cluster. Clustering techniques usually produce less-personal recommendations than other methods, and in some cases, the clusters have worse accuracy than nearest neighbor algorithms [27]. Once the clustering is complete, however, performance can be very good, since the size of the group that must be analyzed is much smaller. Clustering techniques can also be applied as a first step for creating a smaller set of candidates for a nearest neighbor algorithm or for distributing nearest-neighbor computation across several recommender engines [190].

Association rules have been used for many years in marketing, to analyze patterns of preference across products, and to recommend products to consumers based on other products they have selected [190]. Association rules are used to discover regularities between products in large scale transaction data. An association rule expresses the relationship that one product is often purchased along with other products. To select interesting rules from the set of all possible rules, constraints on various measures of significance and interest can be used. The well-known constraints are minimum thresholds on *support* and *confidence*

Association rules are more commonly used for larger populations rather than for individual consumers. For example, they can be used as the basis for decisions about marketing activities such as promotional pricing or product placements in supermarkets. Like other learning methods that first build and then apply models, association rules are less suitable for applications where knowledge of preferences changes rapidly. By contrast, recommender systems based on nearest neighbor techniques are easier to implement for personal recommendation in a domain where user opinions are frequently added, such as e-commerce applications [190].

Latent factor models try to reduce the dimensionality of the space of user-item ratings by mapping users and items to the same latent factor space [109]. The users and items are then related to each other through these latent factors. Examples of latent factor techniques applied to recommendation include Singular Value Decomposition (SVD) [189, 163], factor analysis [42], Probabilistic Latent Semantic Analysis (PLSA) [97]. Factor models have been the most successful recommendation algorithms to date. The successful algorithms in the Netflix competition all used some kind of matrix factorization models [203]. The Netflix prize[1], a research competition with one million dollar prize, was awarded to the research group that could improve the Netflix[2] existing recommendation algorithm for predicting movie ratings by 10%.

Graph-based techniques create a graph from the user-item matrix and use different graph-based appraoches to make a prediction. Horting is a graph-based technique in which nodes are users, and edges between nodes indicate degree of similarity between two users [7]. Predictions are produced by walking the graph to nearby nodes and combining the opinions of neighbors. Horting differs from nearest neighbor as the graph may be walked through other users who have not rated the product in question, thus exploring transitive relationships that nearest neighbor algorithms do not consider [190]. Graph-based techniques have gained more popularity in recommendation in the social Web domain. In chapter 4 we present more details of those approaches.

K-nearest neighbor algorithm is generally considered as one the lazy algorithms in machine learning. However, if all of the user similarities (for user-based CF) or item similarities (for item-based CF) are computed in advance, and not real-time, then we can consider these algorithms as model-based as well. In this case, the model is basically the similarity matrix.

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

[2]www.netflix.com

Advantages and Disadvantages of Collaborative Filtering

The greatest strength of collaborative techniques is that they do not require any information about the product that they are recommending. Thus, they are completely independent of any machine-readable representation of the objects being recommended, and work well for complex objects such as music and movies where variations in taste are responsible for much of the variation in preferences [32]. This feature makes collaborative filtering as the most widely implemented and most popular recommendation technology specifically for e-commerce domains where new products come and go on regular bases.

Another advantage of CF algorithm is that CF algorithms are able to take the quality of an item into account when recommending items, especially in the case of explicit user ratings [22]. For instance, two movies might have the same genre, but might have very different quality. By taking actual user preferences into account, CF algorithms can prevent poor recommendations that are merely based on product features.

Collaborative filtering systems are the only kind of recommender systems that can provide serendipitous recommendation. A serendipitous recommendation is something new, non-obvious and appreciated that the user would likely not have discovered on his/her own. For example, an unfamiliar song from an unfamiliar musician, or a unfamiliar movie from an unfamiliar director.

On the other hand, CF algorithms suffer from several problems. First, collaborative filtering systems are not able to provide explanation for recommendations. Current CF systems are black boxes, providing no transparency or reasoning behind a recommendation. Many users want to know why they get a certain recommendation specifically if there is a high risk involved.

Second problem of CF algorithms is the "cold-start" problem. This problem appears in startup phase of the recommender system, or when a new user or a new item is added to the system. In these cases, the system cannot generate any recommendations. Solutions to this problem include using other data sets to seed the system, using different recommendation algorithms or simply recommending most popular items when the cold start problem exists. Even after acquiring more ratings from the users, sparsity of the user-item matrix can still be a problem for CF.

Another problem mentioned in literature is that collaborative recommenders work best for a user who fits into a niche with many neighbors of similar taste. It does not work well for so-called "gray sheep" [50], who fall on a border between existing cliques of users [32].

2.2.2 Content-based Recommendation

Content-based recommendation algorithms, also known as content-based filtering, can be considered as an outgrowth and extension of the information filtering research. Content-based systems work by creating some kind of representation of the items based on their content features. For example, information retrieval systems like the newsgroup system NewsWeeder [118] use terms in a document as features. A content-based recommender learns a profile of the user's interests based on the features present in objects the user has rated. The type of user profile derived by a content-based recommender depends on the learning method employed. The user profiles might be long-term models that are updated as more evidence about user preferences is observed or might be ephemeral profiles (usually in form of user query).

Typically, content-based filtering is approached as either an IR problem or a machine learning problem [22]. In the IR approach textual similarity (e.g $tf.idf$) is used to match items representations (e.g. documents) against user representations (e.g. query). Examples of this approach include many news recommender applications such as [81]. In the machine learning approach the features of items are used to train a prediction or classification algorithm. Examples of this approach include neural networks used in Re:Agent [23] for email filtering, decision trees and Bayesian networks used in InfoFinder [160] for document recommendation.

Advantages and Disadvantages of Content-based Recommendation

One of the advantages of CB algorithms is that, in contrast to collaborative approaches, they have no cold start problem for new items since the features of the new item can be extracted when the item is added. Second advantage of CB algorithms is that similar to CF algorithms, they do not need deep domain knowledge and it is sufficient to store the item features in a database and collect implicit feedback from the users about their item preferences. However, they are still more explainable than CF algorithms since they match the features that the user is interested in with item features.

The main disadvantage of CB algorithms is that in domains with complex objects such as music and movies where there are large numbers of features, it is difficult to do a perfect content analysis and discover the main real features that make the product attractive to a user. In addition, content-based filtering algorithms can only recommend items from a narrow topic range; they are unable to provide serendipitous recommendations.

2.2.3 Knowledge-based Recommendation

All recommendation algorithms attempt to help a user to find interesting items of their taste. Collaborative filtering algorithms do this based on the behavior of the user and other like-minded users, whereas content-based filtering approaches do this based on the features of the items of users interests and the features of available items. A third class of recommendation algorithms is formed by knowledge-based algorithms. They use domain knowledge, and recommend items based on functional knowledge of how a specific item meets a particular user need [32].

Domain knowledge can take many forms, but without loss of generality we can describe it as an ontology in which there are relations among attributes, objects, and concepts, especially related to item features. It may also contain means-ends knowledge about how items may meet potential user goals. The presence of domain knowledge may permit items to be described with structured representations as opposed to simple vectors of features. A recommender system may also rely on knowledge that is external to the question of product suitability or "best match". The owners of a system may prefer certain recommendations for business reasons. For example, a video rental service may prefer to recommend items from its less-popular back catalog over in-demand new releases. Such knowledge we refer to as business rules knowledge.

Burke [32] identifies three types of knowledge required in a knowledge-based system: catalog knowledge, functional knowledge, and user knowledge. Catalog knowledge is knowledge about the objects being recommended and their features. This type of knowledge is similar to the one used in content-based systems, however, it has more deep knowledge about the relationships between the features. For example, the Entree recommender [40] should know that "Thai" cuisine is a kind of "Asian" cuisine. Functional knowledge is the knowledge that enables the system to map between the users needs and the object that might satisfy those needs. For example, Entree knows that a need for a romantic dinner spot could be met by a restaurant that is "quiet with an ocean view". User knowledge is the knowledge the system gets about the user. This might take the form of general demographic information or specific information about the need for which a recommendation is sought.

Advantages and Disadvantages of Knowledge-based Recommendation

Knowledge-based recommenders are the most reliable approach (if the domain knowledge is comprehensive and up to date) because the background knowledge is free of noise. KB systems, specifically the conversational ones, allow the user to provide a rich specification of their need, which in turn results in more satisfying recommendations. Other advantages of knowledge-based recommendation include no cold start problem, and its ability for intuitive explanations of why a certain item was recommended.

The disadvantage of KB is the high cost and effort for setup and maintenance. It is usually too much of effort for domain experts to capture all the objects, attributes and relations in a domain. As a result, knowledge-based recommendation is not as popular as the other two classes of algorithms.

2.2.4 Other Types of Recommendation

Although in most recommendation literature, the three types of CF, CB, and KB has been identified as main types of recommedation techniques, some researchers have other categorizations. Several researchers such as Adomavicius et al. argue that CB and KB are basically similar since they use item features. Thus, they consider only two main categories of collaborative and content-based techniques [5]. While other researchers such as Burke have considered other categories such as demographic and utility-based as additional recommendation techniques [32]. Burke defines utility-based as systems that make suggestions based on a computation of the utility of each object for the user and demographic recommenders as the ones which categorize the user based on personal attributes and make recommendations based on demographic classes.

Hybrid recommender systems combine two or more recommendation techniques to gain better performance with fewer drawbacks as compared to individual techniques. A survey on existing recommender systems and possible directions in hybrid systems is presented in [32].

2.3 Social Tagging Systems

In the past decade there has been a considerable increase of social websites focusing on facilitating information sharing, interoperability, user-centered design, and collaboration. These websites named as "Web 2.0" allow users to interact with each other and contribute to websites' content, in contrast to websites that users are limited to the passive viewing of information. Examples of Web 2.0 applications include social-networking sites, wikis, blogs, and websites that support content sharing.

An important component of many of these websites is social tagging; allowing users to associate tags to items (also called resources) and share them with others. These items can vary from bookmarks[3], to photos[4], videos[5], books[6], scientific articles[7], movies[8], music[9], slides[10], news articles[11], activities[12], and etc. Tags are freely chosen keywords associated to an item reflecting what the user thinks is the appropriate term to describe that item. Depending on the system's specifications a tag can be made up of one or more words. Some systems like Delicious for example, consider two words separated with space as two distinct tags. In such systems users have to escape the one-word-only limitation by concatenating words. For example, in Delicious "web-design" is one tag but "web design" is considered as

[3]http://www.delicious.com
[4]http://www.flickr.com
[5]http://www.youtube.com
[6]http://www.librarything.com
[7]http://www.citeulike.org
[8]http://www.movielens.org
[9]http://www.last.fm/
[10]http://www.slideshare.net/
[11]http://slashdot.org/
[12]http://www.43things.com)

two distinct tags “web” and “design”. There is typically no limit to the number of tags that may be assigned to a resource and there is no strict hierarchy of tags.

Social tagging systems are gaining more popularity because they offer users several benefits. First, they allow users to organize their own data with a level of freedom not possible in traditional taxonomic filing systems. Second, social tagging systems provide users with the means to openly share this information among friends and colleagues. Third, they also allow anyone to utilize the collective knowledge of others for discovering new topics, resources or perhaps even new friends. Fourth, the reuse of tags creates a dynamic user driven approach to formalize semantic relationships.

Social tagging systems facilitate the retrieval and discovery of resources by providing different possibilities to navigate through the system. Figure 2.3 shows a screenshot of Delicious website showing the popular bookmarks for the tag “ontology”. There are many different options for users to navigate through the search results. On the top, there is an option for recent or popular bookmarks allowing users to look at the most recent or the most popular bookmarks. In front of each link, there is a number showing the total number of people who have saved a specific resource; by clicking on the number, the user can navigate into other users’ profiles and browse through their other tags and resources. The user can also see other tags that have been commonly associated to each resource. On the right side of the screen, “related tags” are presented which are calculated by the system based on the aggregation of user profiles and the number of times two tags have co-occurred together.

Social tagging systems have also been called “collaborative tagging” and “Folksonomy”. Even though some researchers have differentiated between these names, they have been commonly used in the literature as other names for social tagging systems. Bogers [22] differentiates between social tagging and collaborative tagging by considering the two types of tagging systems: broad and narrow systems. In a broad tagging system such as social bookmarking websites, all users can tag the resources while in narrow systems such as photo or video sharing websites only the creator of a resource can tag the item. Bogers considers only the broad tagging systems as collaborative tagging since a tag can be applied multiple times to the same resource in such systems. The word “Folksonomy” is a portmanteau of “Folk” and “Taxonomy” and was first introduced by Vander Wal [215], who defined a folksonomy as the result of “personal free tagging of information and objects for one’s own retrieval”. Different variations on this definition have been proposed in the past. In this thesis, we use the terms social tagging and collaborative tagging interchangeably whereas folksonomy is the outcome of social tagging; a complex network of interrelated users, resources and tags.

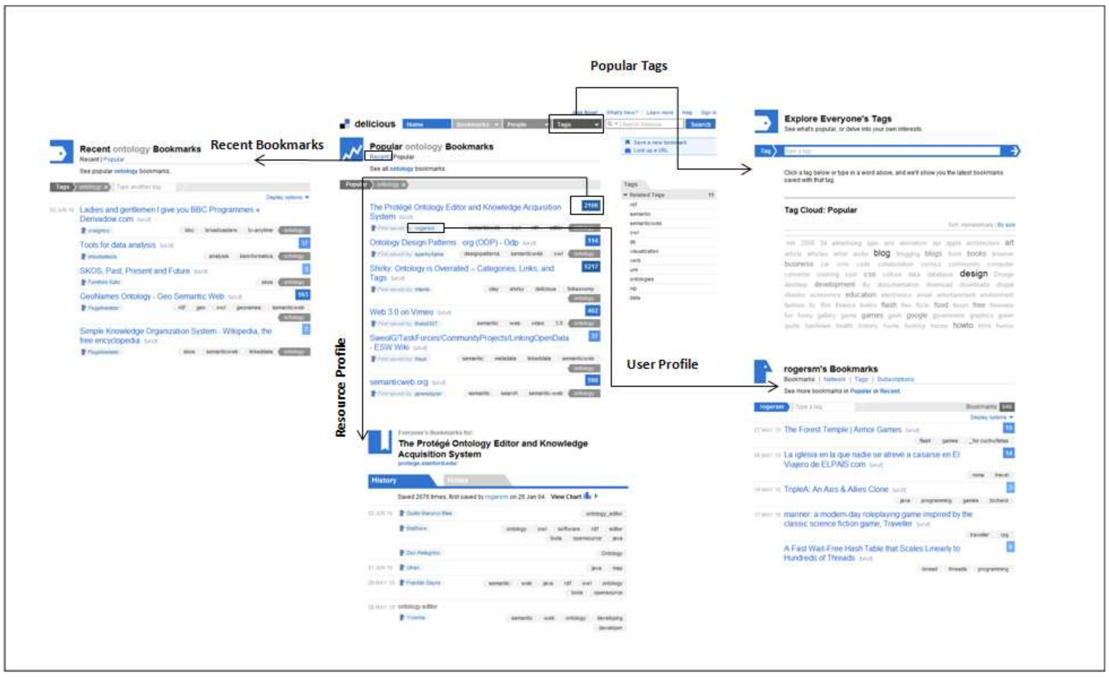


Figure 2.3: Screenshot of Delicious website showing the popular bookmarks for ontology. There are different navigation possibilities through the system.

2.3.1 Challenges in Social Tagging Systems

Despite the many benefits offered by folksonomies, they also present unique challenges. In this section, we briefly discuss some of the major challenges including ambiguity, redundancy, and attacks against social tagging systems.

Ambiguity and Redundancy

Most collaborative tagging applications allow the user to describe a resource with any tag they choose. As a result they contain numerous ambiguous and redundant tags.

Ambiguous tags have multiple meanings. A tag may have different word senses; “apple” can refer to the company or to the fruit. Names may also result in ambiguity; “paris” might mean the city or the celebrity. Subjective tags such as “cool” can result in ambiguity since different users have contradictory notions of what constitutes cool. Finally, overly vague tags such as “tool” can mean gardening implements or software packages. Ambiguous tags can impede users as they navigate the system or burden the user with unwanted recommendations.

Redundant tags share a common meaning: “America” and “USA” confer the same idea. Because users may annotate resources with any tag they choose, folksonomies are full of redundant tags that share a common meaning. Syntactic variance such as “blogs” or “blogging” can cause redundancy. Case (“java” or “Java”), spelling (“gray” or “grey”), and multilinguism (“Photo” or “Foto”) may also result in redundancy. The use of non-alphanumeric characters, abbreviations, and acronyms are other sources of redundancy.

We studied the impact of ambiguity and redundancy on tag recommendation in [69]. In this work, we studied how ambiguity and redundancy can cause misleading evaluation of the effectiveness of tag recommenders. While recommenders are often judged by their ability to predict items occurring in a tests set, the quality of a tag recommender may be underestimated by traditional metrics if it routinely passes up ambiguous tags in order to recommend tags with greater information value. Moreover, evaluation metrics may overvalue a recommender that proposes ambiguous tags despite their lack of specificity. Similarly, redundancy can hamper the effort to judge recommendations as well, but from the opposite perspective. A recommended tag may be counted as a miss even though it is synonymous to a tag in the holdout set. An example may be when the holdout set for a test user contains the tag “java” while the recommendation set contains “Java.” Therefore, redundancy may mask the true effectiveness of the recommendation algorithm: while from the user’s perspective “Java” is a good recommendation, in the evaluation process it would appear as incorrect.

We employed a cluster-based approach to define and measure ambiguity and redundancy. We clustered both resources and tags into highly cohesive partitions based on co-occurrence. A tag is considered ambiguous if several resources from different clusters have been annotated with it. Tags from the same tag cluster are considered redundant. We chose the cluster-based approach over a variety of semantic and linguistic approaches because it provides a more general and language independent method for defining ambiguity and redundancy. We defined metrics, based on the resulting clusters, for measuring the degree of ambiguity for a tag, and the level of redundancy for pairs of tags. We provided extensive evaluation on three real world folksonomies to determine the impact of ambiguity and redundancy across several common tag recommendation algorithms as well as across data sets.

Other researchers have also looked at ambiguity and redundancy problems. Ambiguity is a well known problem in information retrieval and has been identified as a problem in folksonomies as early as 2004 in [134]. Flickr uses the co-occurrence of tags to cluster tags and shows related tags grouped into clusters. Sigurbjörnsson and Zwol [194] use a probabilistic approach to model tag co-occurrences and measure ambiguity to facilitate the tag recommendation for photographs. WordNet has been used to identify ambiguous tags and disambiguate them by using synonyms [119]. Folksonomy search is expanded with ontologies in [106] to solve the ambiguity in tagging systems. In [152] the user profile is used to resolve ambiguity, by considering other tags the user has applied. Clustering was used to measure ambiguity by Yeung et al. [226]. They focus on the network analysis techniques to discover clusters of nodes in networks. The ambiguous tags appear in several clusters and the top ten tags from each cluster are used to find the right context for disambiguation. The work is continued in [227] where a method for exploring the semantics of a tag is described. In [108] multi-dimensional scaling is used for co-word clustering to visualize the relationships between tags.

Entropy as a measure of tag ambiguity has been proposed in [232, 222]. They used a probabilistic generative model for data co-occurrence to determine ambiguous tags. The tagging space is assumed to cover different categories and the tag membership of each category is estimated via the EM algorithm. Our measure of ambiguity in [69] uses a similar approach. We cluster resources and use the distribution of tags across the clusters to measure their ambiguity.

Redundancy in a folksonomy is largely due to the ability of users to tag resources irrespective of a strict taxonomy. In [75] two types of redundancy are identified: structural and synonymic. Structural redundancy refers to terms that are essentially different forms of the same word. Synonymy refers to different tags that share identical or roughly-equivalent meanings. In [214] structural redundancy is explained by stemming to remove suffixes, removing stop words, comparing tags for differences of only one character or identifying compound tags. Synonymic redundancy is evaluated in [9] by using WordNet to determine synonyms. In [67, 68] agglomerative clustering is used to identify similar tags. Clusters of tags are used as intermediaries between users and resources in order to reduce the noise generated by redundant tags. Tag recommendation is one of the techniques to avoid ambiguity and redundancy in social tagging systems. We introduce a graph-based tag recommendation in chapter 4 and we will experimentally show that our algorithm outperforms one of the most successful graph-based tag recommenders, FolkRank.

Attacks Against Social tagging Systems

Like other publicly accessible adaptive systems such as collaborative recommender systems, tagging systems present a security problem. Attackers, who cannot be readily distinguished from ordinary users, may inject biased profiles in an attempt to force a system to adapt in a manner advantageous to them. This problem, even though serious, has not been taken so much of attention in the research community. We discuss the problem in more depth in chapter 6. We will present the dimensions that characterize an attack and outline a framework to identify different types of potential attack strategies against a social tagging system.

In the following sections, we first present our terminology to formally describe a folksonomy and next we review some related work in this area.

2.3.2 Formalization of Folksonomy

In this section, we present our formalization of folksonomies. We use the same formalization in the next chapters. First, we review some basic concepts and definitions in graph theory used in the formalization.

Basic concepts and definitions

We consider the folksonomy as a network and model it with a graph. A graph $G = (V, E)$ is defined with a vertex set V, where $N = |V|$ denotes the number of nodes, and an edge set E. An edge is a two-element subset of V. We interchangeably use terms vertex or node to refer to elements of the vertex set V, and similarly edge or link to refer to elements of the edge set E. We present the definitions of several basic graph-theoretic concepts:
Bipartite graph: Graph G is bipartite if its vertex set can be partitioned into two disjoint sets $V1, V2$, so that there are only edges connecting nodes across the sets $V1$ and $V2$. Or equivalently, there exist no edges between the nodes of the same partition.
Tripartite: Graph G is tripartite if its vertex set can be partitioned into three disjoint sets $V1, V2, V3$, so that there are only edges connecting nodes across the sets $V1$,$V2$ and $V3$. Or equivalently, there exist no edges between the nodes of the same partition.
Directed and undirected graph: A graph is undirected if $(i, j) \in E \Leftrightarrow (j, i) \in E$, i.e., edges are unordered pairs of nodes. If pairs of nodes are ordered, i.e., edges have direction, then the graph is directed.
Hypergraph and Hyperedge: Hypergraph is a generalization of a graph, where an edge can connect any number of vertices. Formally, a hypergraph H is a pair $H = (V, E)$ where V is a set of vertices, and E is a set of non-empty subsets of V called hyperedges.

Folksonomy Graph

In order to model networks of folksonomies at an abstract level, we use the formalization introduced by Mika [138], representing the system as a tripartite graph with hyperedges. The set of vertices is partitioned into the three disjoint sets $U = u_1, u_2, ..., u_k$, corresponding to the set of users, $T = \{t_1, t_2, ..., t_m\}$ the set of tags, $R = r_1, r_2, ...r_l$ the set of resources or objected annotated. In a social tagging system, users associate objects with tags, creating ternary associations between the user, the tag and the resource. Thus we can define A, a set of annotations, represented as user-tag-resource triples.

$$A \subseteq \{\langle u, r, t \rangle : u \in U, r \in R, t \in T\} \tag{2.1}$$

Thus, the folksonomy can be described as a four-tuple D:

$$D = \langle U, R, T, A \rangle, \tag{2.2}$$

To simplify analysis, we can reduce the hypergraph into three bipartite graphs with regular edges. The graphs model aggregate associations between users and resources (UR), users and tags (UT), and tags and resources (TR) [138, 102]. Aggregate projections of the data can reduce the dimensionality by sacrificing information [191]. The relation between resources and tags can be formulated as a two-dimensional projection, RT, such that each entry, $RT(r,t)$, is the weight associated with the resource, r, and the tag, t. This weight may be binary, merely showing that one or more users have applied that tag to the resource, or it may be finer grained using the number of users that have applied that tag to the resource:

$$RT(r,t) = |\{a = \langle u,r,t \rangle \in A : u \in U\}| \tag{2.3}$$

Such a measure is equivalent to *term frequency* or *tf* common in Information Retrieval. Similar two-dimensional projections can be constructed for UT in which the weights correspond to users and tags, and UR in which the weights correspond to users and resource where

$$UT(u,t) = |\{a = \langle u,r,t \rangle \in A : r \in R\}| \tag{2.4}$$

$$UR(u,r) = |\{a = \langle u,r,t \rangle \in A : t \in T\}| \tag{2.5}$$

In words, the bipartite graph UT links the users to the tags that they have associated at least to one resource. Each link is weighted by the number of times the person has used that tag. UR links the users to the resources that they have tagged . Each link is weighted by the number of tags the person has used for that resource. We can further break each bipartite graph into two simple graphs. For example, the UT graph can be folded into two graphs: a social network of users based on overlapping sets of tags; and a network of tags based on overlapping sets of users. Such a network of tags is referred to lightweight ontology of tags by [138]. Figure 2.4 shows the steps of the described process.

The other two bipartite graphs that we derive from the original tripartite model can also be folded into one-mode networks in a similar fashion. In particular, the TR graph leads to another semantic network, where the links between tags are weighted by the number of resources that are tagged with both terms. This type of network mimics the basic method applied in text mining, where terms are commonly associated by their co-occurrence in documents. The UR graph results in another social network of users, where the weight between two users is given by the number of resources they have both tagged. We also get a network of resources, with associations showing the number of people who have tagged a given pair of resources.

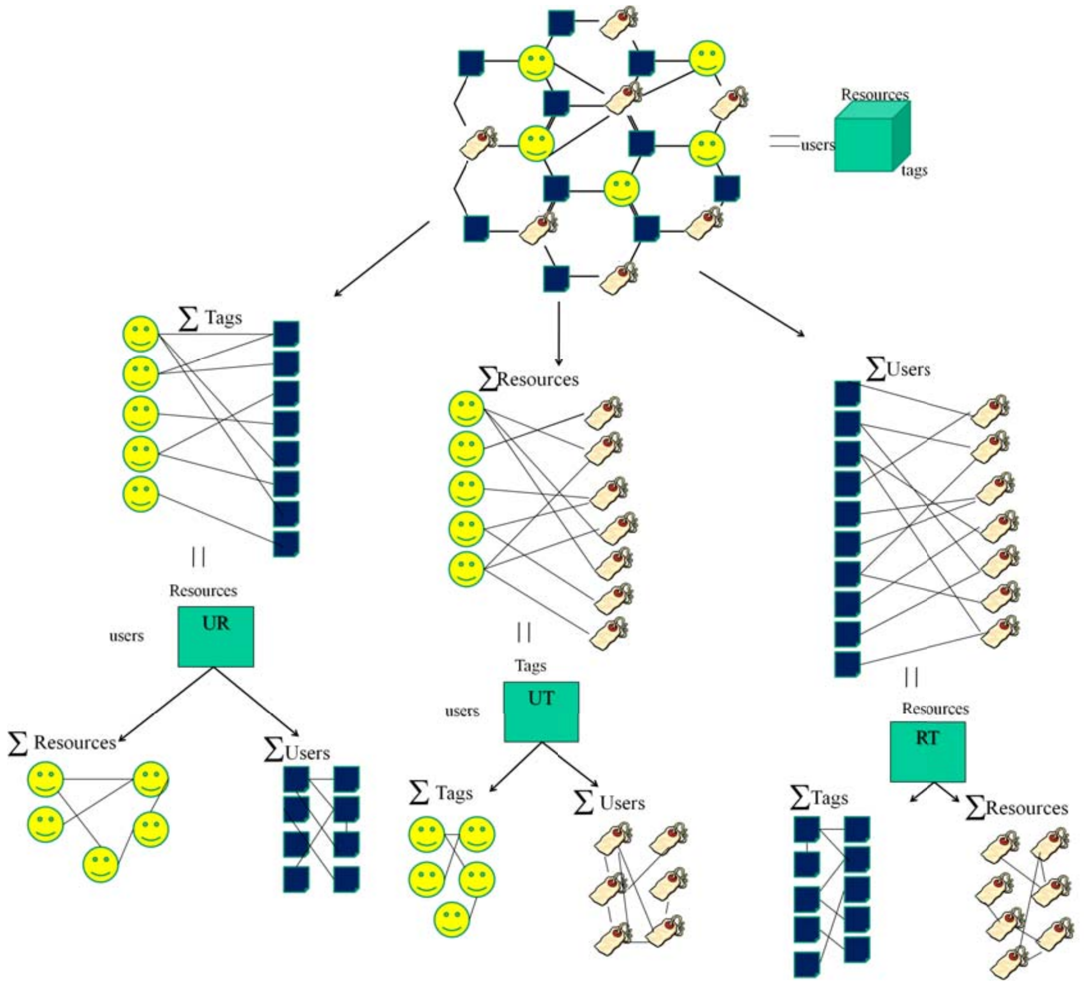


Figure 2.4: We can reduce the hypergraph into three bipartite graphs with regular edges. The graphs model aggregates associations between users and resources (UR), users and tags (UT), and tags and resources (TR)

2.4 Related Work in Social Tagging Research

As social tagging applications have gained in popularity, researchers have started to explore and characterize the tagging phenomenon. In this section we provide an overview of the state of the art in the research area of social tagging systems. First, we review general related work in this area including approaches to characterize different social tagging systems and user motivations. Next, we review different data mining approaches for improving search and navigation in social tagging systems including different dimensions of recommendation. Finally, we present some current research efforts on bridging the gap between ontologies and folksonomies.

2.4.1 Characterize Social Tagging Systems and User Motivations

Many researchers have started to characterize social tagging systems and explore the reasons for emergence and popularity of tagging phenomenon [124, 76]. One of the significant formal studies of tagging systems appeared in the work of Golder and Heberman [76]. The authors studied the information dynamics in collaborative tagging systems, specifically, the delicious system. The authors discussed how tags have been used by individual users over time and how tags for an individual resource stabilizes over time. They also discussed two semantic difficulties: polysemy (when a single word has multiple related meanings) and synonymy (when different words have the same meaning). Macgregor and McCulloh provide an overview of the phenomenon and explore reasons why both social tagging and ontologies will have a place in the future of information access [124]. Ames and Naaman investigate the incentives and motivations for tagging in photo sharing systems such as Flickr. They present a taxonomy of motivations which combines sociality and functional motivations [10].

Chi and Mytkoswicz [47] have analyzed Delicious and found that the efficiency of social tagging decreases as the communities grow; that is, tags are becoming less and less descriptive and consequently it becomes harder to find a particular item. Simultaneously, it becomes harder to find tags that efficiently mark an item for future retrieval.

Adam Mathes [134] describes tags as user-created meta-data, where users of the documents and media create meta-data for their own individual use that is also shared throughout a community. Mathes points to several limitations and strengths of tagging systems. The major limitations include ambiguity, synonymy, and multiple words. The multiple words problem happens because most tagging systems are designed primarily to deal with a single word as a tag. So multiple word tags are subsequently parsed as separate tags (”Bill” and ”Clinton”); or users escape the one-word-only limitation by concatenating words, for example “Bill-Clinton” or “BillClinton”. All these different variations can make the search and filtering of the system inefficient. We will discuss the ambiguity and redundancy problems in more detail in section 2.3.1. The strength of tagging systems that Mathes points in [134] are serendipity while navigating through the system and the freedom of users to use their individual vocabulary. In a similar study, Wu et al. discuss the benefits and challenges of social tagging systems [221]. They propose folksonomies as potential technological infrastructure to support knowledge management activities in an organization or a society. However, they also point to the problem of lack of a document hierarchy in the knowledge taxonomy emerged by employee-generated folksonomy which prevents it from being widely adopted by enterprises. The authors suggest harvesting social knowledge by identifying communities of common interest, and identifying information leaders or domain experts, and generating ontologies.

Marlow et al. [133] analyze and compare the design and features of existing social tagging systems and develop two organizational taxonomies for social tagging systems . The first taxonomy describes system design and attributes and the second taxonomy describes user incentives. The first taxonomy is presented in seven dimensions including tagging rights (self-tagging, permission-based, free-for all), tagging support (blind, suggested, viewable), aggregation model (bag, set), resource type (textual, non-textual), source of material (user-contributed, system, global), resource connectivity (links, groups, none), and social connectivity (links, groups, none). The second taxonomy describes motivations of tagging and is categorized into two high-level categories of organizational and social.

2.4.2 Resource Recommendation and Personalized Search

Many researchers have explored different data mining techniques to facilitate user navigation, and to improve search and personalization in social tagging systems.

Recommender systems have been widely applied in folksonomies to overcome information overload and to help users to find high-quality sources, whether resources (e.g documents) or people. Unlike traditional recommender systems which have a two-dimensional relation between users and items, tagging systems have a three dimensional relation between users, tags and resources. Recommender systems can be used to recommend each of the dimensions based on one or two of the other dimensions.

Resources in social tagging systems can vary from bookmarks to video, music, photos and etc. Social tagging systems are potentially a suitable place for people interested in a specific domain to find new resources of their interest. However, to deal with the amount of information, users need to get supported by the system. Helping users to find interesting resources can be considered as "search", when a user explicitly types a query, or "recommendation", when the system suggests a resource to a user based on their profile and activities without the user explicitly asking for it. The search results might be personalized, in which the user profile such as user tags and resources are taken into account. In a personalized search, each user might get a different output for the same query. In this subsection, we review the state of the art in both areas of personalized search and recommendation of resources in folksonomies.

Traditional item recommendation algorithms can be used to recommend resources to users. However, tags can enrich user profiles with additional valuable information for recommendation. For instance, authors in [206] apply user-based and item-based collaborative filtering to recommend resources. The authors incorporat tags into standard CF algorithms by reducing the three-dimensional correlations to three two dimensional correlations and then applying a fusion method to re-associate these correlations. Similarly, [151] and [150] use tags as context information to recommend resources. Similar approaches for search personalization using user tags are presented in [223, 209]

Using tag clustering for resource recommendation is presented in [153]. In this work, first an affinity level between a user and a set of tag clusters is calculated. A collection of resources are then identified for each cluster based on tag usage. Resources are recommended to the user based on the user's affinity to the clusters and the associated resources. Similarly, the role of tag clustering in personalized search and navigation is presented in [72, 71, 193]. The authors show that clustering provides a means to overcome redundancy and ambiguity thereby facilitating recommendation. Similar clustering algorithm for search personalization in suggested in [18]. In this work, the latent semantic associations between tags is captured using a third-order tensor technique. Then, hierarchical agglomerative clustering was employed to discover relevant resources and rank the search results. Pan et al. [156] suggest expanding folksonomy search with ontologies to address the problem of ambiguity in tagging systems. Probabilistic models have been utilized in [162] and [217] for resource recommendation.

In [100], a novel algorithm, FolkRank, for search and ranking in folksonomies is proposed that considers the interrelations among tags, resources and users. The authors extend the commonly known PageRank algorithm to folksonomies under the assumption that users, resources and tags are important if they are connected to other important tags, resources and users. They use a weight passing scheme to derive the importance of an object in folksonomies.

In addition to personalized search within the folksonomies, researchers have proposed to use social bookmarking websites such as Delicious for improving search results in search engines. In [14], ranking of web search is optimized using social annotations by taking into account the similarity of the query to the Webpages in Delicious. Morrison [149] compared the search performance of social bookmarking Websites against search engines and subject directories and found out that folksonomy search results overlap with those from the other systems, and documents found by both search engines and folksonomies are significantly more likely to be judged relevant than those returned by any single IR system type.

2.4.3 Tag Recommendation

Tag recommendation, the suggestion of tags during the annotation process reduces the user effort. Tag recommenders assist the tagging process by suggesting users a set of tags that they are likely to use for a resource. Personalized tag recommenders take the users' tagging behavior in the past into account when they recommend tags. Researchers have applied different data mining and machine learning techniques to the tag recommendation problem.

A comparison of user-based collaborative filtering and a graph-based recommender based on the PageRank algorithm to recommend personalized tags is offered in [105]. Association rules are explored in [94] to recommend tags and introduce an entropy-based metric to find how predictable a tag is. The title of a resource and the user vocabulary is used in [122] to generate recommendations. The results show that tags retrieved from the user's vocabulary outperform recommendations driven by resource information. The authors in [224] present general criteria for a good tagging system including high coverage of multiple facets, high popularity and least-effort. They categorize tags into different categories of content-based, context-based, attribute, subjective, and organizational tags and use a probabilistic method to recommend tags. Basile et al propose a classification algorithm for tag recommendation in [15] and Adrian et al. suggest a semantic tag recommendation system in the context of a semantic desktop in [6].

User-defined tags and co-occurrence are employed by [194] to recommend tags to users on Flickr. The assumption is that the user has already assigned a set of tags to a photo and the recommender uses those tags to recommend more tags. A similar study is conducted in [66] and a classification algorithm for tag recommendation is introduced. We adapted K-Nearest Neighbor for tag recommendation in [70] and showed incorporating user tagging habits into recommendation can improve recommendation.

Increasing interest in improving the effectiveness of tag recommendation attracted many researchers from all over the world to the ECML/PKDD Discovery Challenge 2009 [60] which was mainly focused on tag recommendation. Different data mining approaches including content-based techniques, probabilistic models, and factor models have been applied for tag recommendation. The comparison of different approaches [174] shows that graph-based and factorization models have the best performance in tag recommendation.

Most of the mentioned approaches for tag recommendation are appropriate for dense parts of the data where there is enough information available about the user or the resource. Thus, these approaches have problem in cold start situation when a new user or new resource is introduced into the system, or if the resource has not been associated to certain number of tags by certain number of users in the past. In these cases, content-based approaches are more applicable. Song et al. utilized machine learning algorithms including gaussian process classification, SVM model, vector space model and poisson mixture model to predict tags based on the content [197, 198]. One part of the ECML/PKDD 2009 Discovery Challenge was dedicated to the sparse part of the data to challenge content-based algorithms for tag recommendation. The succefull approaches relied on a combination of good preprocessing, some external knowledge sources and a good heuristic to choose the right set of tags [123]. An extensive comparison of different content-based algorithms for tag recommendation is presented in [104].

2.4.4 User Recommendation

Identifying communities of common interest, and information leaders or domain experts are the major potentinal benefits of social tagging systems from a knowledge management point of view [221]. User recommendation is common in social networking systems in which there is explicit connections among users. For example, facebook recommends new friends based on the number of friends in common. However, the more interesting application is to help users to find domain experts or like-minded people based on their profiles. Some social tagging systems such as Last.fm show users their neighbors, the people that have similar listening tastes. In such scenarios, users can find new friends or groups with similar taste who they did not know before; this recommendation can be more useful since it is more serendipitous and provides users with a valuable information that they could not easily find out by themselves.

ExpertRank is introduced by [107] to quantify users' expertise in the context of a tag in an enterprise tagging system. Two models are proposed to calculate ExpertRank. The first model simply assumes an unstructured tag space where tags have no dependencies and expertise gained in a tag context is independent of expertise gained in another tag context. The second and more realistic model assumes clustered tag space where each cluster contains set of tags related to each other. The relationships between the tags in a cluster are represented by links between tags and indicate overlapping expertise areas. The model enables a mechanism similar to the personalized version of the PageRank algorithm [29] to propagate expertise through the linked structure of the tags.

Another approach to expert finding is introduced in [64]. In this work, user-provided content and taxonomy classifications are used to describe topics of expertise. The approach is implemented for the DBLP[13]-Expert Finder, a system for expert finding and exploratory search across researchers and topics in the field of computer science. Similar approach to expert finding is presented in [41]. The paper describes an approach that finds experts, expertise and collaboration networks in the context of the peer-review process. A taxonomy of computer science topics and an ontology of publications is used to create relationships from papers to one or more topics based on paper abstracts and keywords. Next, an expertise

[13]BDLP:http://dblp.uni-trier.de/

profile for a researcher is built based on the aggregation of the topics of his/her publications. Tag-based profiles are used in [58] to find persons with similar interests. Similar to previous ones, this work also uses DBLP collection as data set and the paper keywords as tags. Although a scientific database such as DBLP has some similarities to a narrow social tagging system, it has different properties. Such data is free of noise and has key features such as the abstract or citations that can help identify the properties of resources precisely.

The literature focusing on the problem of user recommendations in folksonomies is still sparse. One of the bottlenecks might be the difficulty of evaluating such recommendation in comparsion to tag or resource recommendation.

2.4.5 Emergent Ontologies from Folksonomies

Peter Mika [138, 139] is one of the first researchers who proposed social tagging systems as a semantic social network which could lead to ontology emergence. The idea roots in the emergent semantics proposed by [2] and the vision is a community of self-organizing, autonomous, agents co-operating in dynamic, open environments, each organizing knowledge (e.g. document instances) and establishing connections according to a self-established ontology. Mika suggests an Actor-Concept-Instance model of ontologies using the semantic-social networks in the form of a tripartite graph of person, concept and instance associations, and extends the traditional concept of ontologies (concepts and instances) with the social dimension. He suggests reducing the hypergraph into three bipartite graphs with regular edges. These three graphs model the associations between actors and concepts, concepts and instances, and actors and instances which are the same as UT, TR, and UR explained in section 2.3.2. Similarly, [210] proposes deriving ontologies from folksonomies by integrating multiple resources and techniques. These resources are: (1) the statistical analysis of folksonomies (2) online lexical resources like dictionaries, Wordnet, Google and Wikipedia (3) ontologies and Semantic Web resources (4) ontology mapping and matching approaches, and (5) involving the community.

The idea of ontology of Folksomony is also proposed by [78] and [59]. Thomas Gruber in [78] discusses the differences of ontology and folksonomy and points out some design considerations for constructing ontologies from tags. The authors in [59] provide more details on the ontology model and an algorithm to create such an ontology from a folksonomy. Their model consists of an ontology designed in OWL that defines the following classes: Source, Resource, Tag, User, Annotation, AnnotationTag and Polarity. For the class Tag, two subclasses are also defined: "TagPersonal" and "TagCommon" which are used to classify the existing tags according to their type, separating the ones of personal type, like those related to the planning of personal tasks or self-reference tags such as"to-read" from the rest of tags (TagCommon). The tag class has the properties of "hasAltLabel" and "hasHidden-

Label" which are meant to represent the different variations of a tag, including singular and plurals, verbal tenses, synonyms, misspellings, incorrect syntactic forms, etc., from the tags preferred representation. For example, the tag with preferred value "New York", could have associated to hasAltLabel the strings "new-york", "york", and to hasHiddenLabel the strings "neyork", "nyc", etc. The authors suggest an algorithm for creating the ontology from the folksonomy which basically covers the associations between user, resources and tags but does not include how to find the tag types or properties. Although the idea of separating the tag types and the tag properties seem interesting, no well-defined algorithm is presented for finding such relations. In a similar work, [234] suggests mapping tags to an ontology and presents the process of mapping in a simple example. However, this work also postpones the automatic creation of an ontology and mapping tags to the related class of the ontology for future research.

Other researchers have started solving the details of the problem using information retrieval and machine learning techniques. Heymann and Molinay [93] suggest creating a hierarchical taxonomy of tags. Their algorithm calculates the similarity between tags using the cosine similarity between tag vectors and each new tag added to the system will be categorized as the child of the most similar tag. If the similarity value is less than a pre-defined threshold then the new tag will be added as a new category, which is a new child for the root. The problem with this algorithm is that there is no heuristic to find the parent-child relation. Any new similar tag will be considered as a child of the most similar tag previously added to the system although it might be more general than the other tag.

Markines et al. [132] present different aggregation methods in folksonomies and present different similarity measure for evaluating tag-tag and resource-resource similarity. Authors evaluate their approach by comparing the results with WordNet for tag similarity and ODP[14] for resource similarity. Although this work presents a strong grounding for finding related or similar tags in folksonomies, it does not provide approaches for finding the kind of relationship between tags such as super or sub-concept relations.

Marinho et al. [13] use frequent itemset mining for learning ontologies from Folksonomies. In this work, a folksonomy is enriched with a domain expert ontology and the output is a taxonomy which is used for resource recommendation. Authors use two major heuristic to create the taxonomy. First, more popular tags are considered more general and second, a tag x is a super-concept of a tag y if there are frequent itemsets containing both tags such that $support(x) > support(y)$.

Hotho et al. suggest using association rules for discovering semantic relations beteen tags [99]. Wu et al. use probabilistic EM algorithms to estimate the probability of co-occurrences of folksonomy elements to obtain the emergent semantics contained in the data [232, 222]. The discovered semantic relations are then applied for semantic search and re-

[14]Open Directory Project

source discovery. Similarly, Zhou et al. [235] propose a probabilistic clusetring approach named Deterministic Annealing (DA) for extracting hierarchical semantics from social annotations.

An ontology learning approach that captures the hierarchical semantic structure of folksonomies is proposed in [204]. The proposed approach consists of three stages. In the first stage, generative probabilistic models are used to model correspondences between tags and documents. The basic idea is that assuming each tag has multiple submeanings, a tag having similar high distributions on multiple topics indicates a high likelihood of being a general tag; while a tag having a high distribution on only one specific topic indicates that the tag possibly has a specific meaning. In the second stage, the possible relations between tags are estimated. Four divergence measures between tags are defined based on the modeling results from the previous stage in order to quantitatively characterize the possible relations between tags. In the third stage, the relation between tags are determined and a hierarchical structure is constructed.

Although many tools have been developed in recent years to support automatic ontology creation, the resulting ontologies are still not meant to be used directly by the end ontology users and the construction of ontologies continues to be a manual, labor-intensive exercise carried out by specialists in the domain [216, 126]. In chapter 5, we propose a semi-automatic approach to support users to collaboratively develop an ontology in a social Web environment.

2.5 Chapter Summary

In this chapter, we gave an overview of the state of the art of recommender systems and social tagging systems. We presented different recommendation technologies along with their advantages and disadvantages. Next, we presented social tagging systems and our terminology to model a folksonomy which will be used throughout this thesis. Next, we focused on the related work in the social tagging area. We introduced different possible recommendation tasks in social tagging systems and briefly reviewed current state of the art. We also presented the current research efforts on creating ontologies from folksonomies.

In this chapter we discussed some major challenges that social tagging systems suffer from including ambiguity, redundancy, and attacks against social tagging systems. Our next chapters present our efforts on addressing these challenges.

Chapter 3

Matching Recommendation Technologies and Domains

3.1 Introduction

This chapter presents a comprehensive taxonomy of recommender systems with guidelines on how to select and apply these systems in different domains. In this chapter we look at recommender system from a broad perspective. Unlike the common traditional view on recommender systems as being used only in e-commerce applications [190], we consider a broad definition of recommendation systems that captures several domains. We consider recommender systems as ones that enable a particular kind of interaction with the user: "any system that produces individualized recommendations as output or has the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options" [36]. This expansive definition makes the scope of recommender systems research quite broad, but it fails to give much guidance on what criteria to consider for implementing such systems.

With the rapid development of recommender system technologies in the recent years, it has become difficult for developers to determine which technology is suitable to a particular context. To alleviate this difficulty, we propose a framework that organizes the space of recommendation problems and provides a systematic approach to finding the appropriate recommendation technology for addressing a given problem in a specific domain. A crucial question this chapter tries to address is how recommendation techniques can be matched to recommendation domains.

In this chapter, we first distinguish among different recommendation algorithms based on the source of knowledge they use. Based on existing literature, we classify the source of knowledge in recommender systems to three different kinds: individual knowledge which basically comes from user input in different forms, social knowledge which is the aggregation of masses of individual knowledge, and content knowledge which can be in form of item features or can get a more complex form of domain knowledge.

Next, we extract the domain characteristics that can help an implementer to select the suitable recommendation technology. These characteristics include heterogeneity, risk, churn, interaction style, preference stability, and inscrutability. We describe each characteristic and explain how they can effect the knowledge sources. Finally, we map the domain properties to recommendation technologies and provide examples from the real world application and recommender system literature.

The research approach for this study consists of a meta-analysis of the research literature and real-world applications. More than 70 articles and application examples of recommender systems from the 12-year period 1997-2008 were gathered for analysis. This time frame covers the development of recommender system research from its early stages to the present time and is extensive enough to identify the different domains that recommender systems have been applied to. The articles then were categorized based on their domain of recommendations. Main application domains include e-commerce, movie, music, news, Web page, real financial service recommendation, etc. Next, we tried to identify the main characteristic of each domain that influence on the selection of the appropriate recommendation technology. We investigated the impact of each characteristic on the knowledge sources of recommendation and mapped those characteristics to the recommendation technologies based on the knowledge sources they use.

This work can be useful for researchers studying recommender systems and implementers considering applying them. The taxonomy provides a useful initial framework within which researchers can place their work. For implementers, the chapter provides a means of making choices among the available technologies. Implementers can use our framework to choose the suitable recommendation algorithm for their specific problem based on the domain characteristics.

3.2 Related Work

As recommender system as a research field has been receiving more attention from academia and industry in recent years, researchers have developed several taxonomies for recommender systems to analyze and classify these systems.

Burke [36] distinguishes between five different recommendation techniques: collaborative, content-based, utility-based, demographic, and knowledge-based. He discusses the advantages and disadvantages of each technique and proposes hybrid recommender systems to gain better performance with fewer of the drawbacks of any technique in isolation. An early work in recommender systems [176], which focuses on collaborative recommenders, introduces a taxonomy of recommender systems based on 5 dimensions. The dimensions characterize properties of the users' interactions with the recommender and the aggregation methods of users' evaluations (ratings). These dimensions include (1) Contents (output) of the recommendation which can be anything from a single bit (recommended or not), numeric rating, a document or URL, etc. (2) Type of user input (implicit or explicit) (3) degree of privacy of the recommender system (4) Aggregation technique (different variations on weighting the ratings) (5) Kind of presentation of the recommendations (sorted according to numeric evaluations, negative recommendations filtered out, etc).

Konstan and Schafer [190] present a taxonomy of collaborative e-commerce recommender applications that separates their attributes into three categories: the functional input/output (targeted customer input, community input, recommendation output), the recommendation method (raw retrieval, manually selected, statistical summarization, attribute-based, item-item correlation, user-user correlation), and other design issues such as degree of personalization and delivery methods. Delivery methods include "push" (email, sending promotional offers), "pull" which allows the customer to control when recommendations are displayed, and "passive" delivery which presents the recommendation in the natural context of the rest of the E-commerce application. Examples of passive recommendation include displaying recommendations for products related to the current product in Amazon.

An eight-dimensional taxonomy of recommender systems is presented in [146]. The dimensions group in two blocks: dimensions regarding profile generation and maintenance and dimensions related to profile exploitation. The authors then carry out a cross-dimensional analysis among various recommender systems in several domains and detect common patterns in the recommenders of the same domain.

A more recent survey on recommender systems [5] classifies recommendation methods (omitting knowledge-based) into three main categories: content-based, collaborative, and hybrid recommendation approaches and classifies recommenders in each category into either heuristic-based or model-based. This study also describes limitations of current recommendation methods and discusses possible extensions to existing systems including improvement of understanding of users and items, incorporation of the contextual information into the recommendation process, support for multi-criteria ratings, and a provision of more flexible and less intrusive types of recommendations.

The work in this chapter is distinguished from previous categorizations in that it is not aimed at classifying existing recommender systems along particular dimensions of interest as in the surveys above. Instead, we look at the recommendation problem from a different angle considering the domain properties. We focus on the problem characteristics and we aim at giving a guideline for selecting among technologies based on these characteristics. We outline the knowledge sources required for recommendation and the constraints related to them and discuss how domain characteristic would influence knowledge sources and the technologies. The chapter discusses the applicability of different recommendation techniques to different types of problems and aims to guide decision making in choosing among these techniques. As such, it might be considered to serve as a sort of recommender for recommender system implementers.

3.3 Knowledge Sources

A recommender system, similar to any other system, has a set of input and output. The input come from user interactions with the system and the output is the recommendation that can be in different forms (e.g. list of top *n* recommendations or predicted rating for items). For the purpose of this chapter, a recommender system consists of two main parts: a knowledge source and an algorithm to generate recommendations using those sources. The choice of algorithm in many cases would depend of the kind of knowledge sources available in the system.

For an individual instance of recommendation, we are presented with a particular target user and our goal is to make recommendations to him or her. In order to have personalized recommendations, a recommender must have knowledge of its target user, called "individual knowledge" in this chapter. In addition to knowledge about the target user, a recommender system needs sources of knowledge to generate recommendations. Therefore, in addition to the individual knowledge, we consider two types of knowledge sources that may come into play in recommendation.

- **Social:** Knowledge about the larger community of users other than the target user.
- **Content:** Knowledge about the items being recommended and/or the domain of recommendation, including how recommended items are used and what needs they satisfy.

Our taxonomy of knowledge sources is adapted from the taxonomy presented by Felfernig and Burke in [61]. In the following subsections, we briefly describe the types of knowledge source, expanding each category into subtypes of knowledge, all of which have been used in some existing recommender systems.

3.3.1 Individual Knowledge

Individual knowledge comes from the input of the target user into the system and it constructs the user profile. User profile might be ephemeral or persistent. Persistent user profiles accumulate user inputs from multiple interactions over time while ephemeral user profiles only gather input from the current user interaction.

User input may be relatively implicit, in the sense that the user does not intentionally provide input for receiving recommendation but interacts with the system as part of some other application, or it may be explicit in that the user explicitly specifies his or her interest before or during the recommendation process.

Figure 3.1 shows the taxonomy of knowledge sources including different kinds of individual knowledge. [61] identifies three main categories of individual knowledge including opinions, demographics, and requirements. We add the user behavior as another category. User behavior is commonly considered as implicit input and it includes Web navigation, items added to shopping cart, user transactions, request for information, etc. User opinions is usually explicit and it includes ratings, reviews, and tags. Demographic information might be explicitly entered by the user or be implicitly inferred by the system and it includes region, age, sex, etc.

User requirements include queries, constraints, preferences, and context. A query is a request from the user and may be formulated as a natural language request, spoken dialog, key words, parameters chosen from a menu, or an example of a similar item. Constraints are user restrictions and limitations that the recommender system must take into account. There is no acceptable solution that violates user constraints. For example, a person may be looking for only German language movies. A preference is a softer constraint, something that the user would prefer in a solution but a solution that violates it might be acceptable. For example, a person may have a preference for a three bedroom apartment, but might be satisfied with a two bedroom if it is a particularly good deal. The users context consists of the external circumstances associated with the recommendation or the users situation. For example, the users location might be an important contextual factor in a restaurant recommendation, with closer establishments being preferred. The use of context in different types of recommender algorithms including collaborative recommendation is an area of active research [4, 165, 112, 17, 212, 228]. However, there is no consensus on how best to benefit from it or even how to define the term.

3.3.2 Social Knowledge

Social knowledge is the total sum of all of the user profiles stored in a system. Since the social knowledge is the aggregation of all individual knowledge in the system, the relationship between individual and social knowledge can be reciprocal. In the sense that the target user opinion is considered as individual when the system is giving him a recommendation, but social when another user relies on them.

Collaborative recommendation is intensive in its application of social knowledge, usually

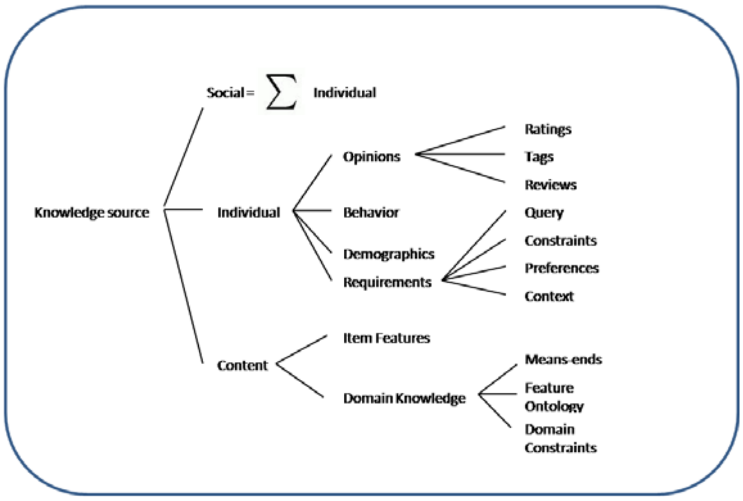


Figure 3.1: Taxonomy of knowledge sources in recommendation

with profiles of the simplest type: user opinions such as the liked-disliked scales used in MovieLens and other well-known collaborative recommenders, or interaction histories as seen in collaborative Web personalization [54].

Ratings are the most straightforward type of user input used to model social knowledge. Rating knowledge is often conceptualized as a $m \times n$ matrix where m is number of users and n is the number of items and each entry corresponds to a user's rating of an item. Model-based techniques use this matrix to create a model in advance, whereas memory-based techniques use it at the time of recommendation generation to produce the prediction.

The use of other types of user opinion is an area of active research. User tags are a promising source of opinion knowledge for recommendation [153, 85, 131]. In this case the data will be a tripartite graph of user, item and tag. More detail is provided in section 3.9. Textual data in the form of reviews have also been used as social knowledge for recommender systems, for example in [3].

In addition to user opinions, aggregation of other types of user input such as behavior, demographic, and requirements can form social knowledge. In the I-SPY collaborative Web search application, user's queries (requirements) are recorded as well as their preferences (link selections) relative to those queries [196]. Demographic information is also employed by some recommender systems. Krulwich [115] uses demographic groups from marketing research to suggest a range of products and services. Similarly, [161] uses machine learning to train a classifier based on demographic data.

3.3.3 Content

Content knowledge has a variety of forms. In its simplest incarnation, the system might only have knowledge about the features of items that it is recommending, enabling it to learn what features a user seems to prefer. The item features may be a simple set of attribute value pairs, such as might be found associated with a product in a database, or the item description may itself be structured as in the case of complex products such as a computer [61].

If items are represented by unstructured documents such as news stories, the implementer will need to draw from information extraction (IE) techniques to extract and select features for use in recommendation. Standard techniques include eliminating stop words and stemming to simplify the feature space. Features can be reduced further by applying more sophisticated feature selection techniques such as information gain, mutual information, cross entropy or odds ratio [141]. Applications of IE techniques to extract content knowledge from semi-structured and structured documents are discussed in [110].

Domain Knowledge

Domain knowledge typically has richer knowledge about the items and the domain than just the features of the items. One common form of domain knowledge that a recommender can employ is a domain ontology over the item features. Such an ontology allows the system to reason about the relationship between features at a level deeper than just raw equality or difference. For example, the restaurant recommender Entree [37] has an ontology of different types of cuisine and can determine that a Thai restaurant would be more similar to a Vietnamese restaurant than a German one would be.

Domain constraints may be necessary to prevent a system from recommending an item that is inconsistent with what the domain permits. For example, a particular insurance policy may only be available to non-smokers. The recommendation problem can be in some cases formulated entirely as constraint satisfaction with constraints being contributed both by the user and by the system.

A final category of domain knowledge is means-ends knowledge, which is the knowledge that enables a system to map between the user's goals (ends) and the products that might satisfy them (means) [61]. For example, a camera buyer might not know much about digital cameras, but he might know that he wants to take photos of his daughter's basketball games. Part of the reason that users benefit from recommender systems is that they can make good choices without necessarily being conversant with all of the complexities of the product space.

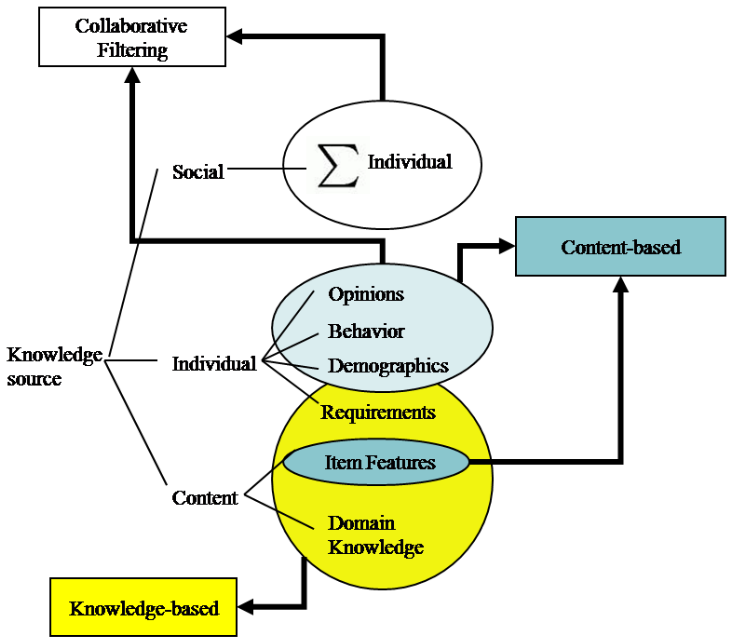


Figure 3.2: Knowledge sources and recommendation types

3.4 Recommendation types

Recommendation types are explained in detail in chapter 2. However, in conjunction with a discussion of knowledge sources, it is worth considering how different recommendation types operate.

- Collaborative recommendation matches an individual knowledge source with a social knowledge source of the same type and extrapolates the target user's preferences from his or her peers.
- Content-based recommendation on the other hand is individually-focused, using item features and user input to learn a classifier that can predict user preferences on new items.
- Knowledge-based recommendation is more of a catch-all category in which the recommender applies any kind of domain knowledge more substantive than item features.

Figure 3.2 shows the connection between knowledge sources and the recommendation types. In this figure, we have included the three main categories of recommendation in the literature. Hybrid recommendation is a matter of combining knowledge sources that have not traditionally been put together in the three types discussed above. Often, a hybrid is created by adapting an algorithm for one recommendation type to accept a knowledge source more typically associated with another type.

As we mentioned before, a knowledge source does not make a recommender system. The system also needs algorithms. It is difficult to generalize since new recommendation algorithms are put forward with great regularity, but in general, collaborative systems use multi-class classification algorithms for extremely sparse and high-dimensional spaces; content-based systems use binary learning algorithms for lower-dimensional spaces; and knowledge-based recommenders use inference schemes of various types. For more details on recommendation algorithms, please refer to 2.2.

An algorithm can only function with the right knowledge sources, and it is on this topic that this chapter will concentrate. Considerations about the domain of application and the style of interaction with the user lead us to conclusions about the availability and characteristics of different knowledge sources. These considerations in turn can be used to guide the selection of feasible recommendation algorithms. We turn next to the characteristics of domains.

3.5 Domain Properties

A domain of recommendation is the set of items that the recommender will operate over, and may also include the set of aims or purposes that the recommender is intended to support. A specialized recommender, for example, a news recommender that identifies stories for the attention of government intelligence analysts, may have different implementation considerations than a generalized news recommender such as Google News. In turn the characteristics of the domain affect the availability and utility of different knowledge sources. In the online news case, there are a huge number of news sources and articles such that no user will ever have time to experience or rate more than a small fraction of them. In addition, the news itself is undergoing constant change. So, we can characterize the "social opinions" knowledge source as one of great sparsity and great dynamism.

Another aspect of the domain has to do with the larger application in which the recommender is embedded. If the recommender is a part of a larger system like an e-commerce site, it may be necessary for the recommender to impose as little as possible on the normal user interaction with the application, which means the system has to use implicit user inputs. On the other hand, if the recommender is the primary application that users are accessing, it can gather explicit data from users.

The properties of the domain affect the choice of knowledge source and recommendation algorithm. Our goal is to identify the most important characteristics of the domain that impact on the selection of proper recommendation technology. We have identified six important characteristics of the domain: heterogeneity, risk, churn, interaction style, preference stability, and scrutability.

3.5.1 Heterogeneity

A heterogeneous item space encompasses many items with different characteristics and different goals they can satisfy. For example, an e-commerce recommender system as found at Amazon.com has a large number of heterogeneous items that can be recommended. Even within a single category like books, such disparate categories as home repair, romance novels, cooking, and children's fantasy all coexist in the database.

On the other hand, a homogeneous recommendation domain has items of the similar type. It is easier to acquire and maintain content knowledge in a homogeneous recommendation space. Consider a site that only recommends digital cameras versus one that has all kinds of electronics. The camera-only site would be able to invest in content knowledge specific to photography, whereas the general site would have a much more challenging task trying to do knowledge engineering for all of consumer electronics. Even a simple catalog of item features becomes difficult to design effectively if the items differ wildly from each other.

3.5.2 Risk

Recommendation domains can be distinguished by the degree of risk that a user incurs in accepting a recommendation. A 10 $ music track is low risk; a $1.5 million condominium or a medical diagnosis could be very high risk. Risk determines the user's tolerance for false positives among the recommendations. In some domains, false negatives may also be important – if there is a cost or risk associated with not considering some options.

Another way to think of a high-risk domain is that there are likely to be some important constraints on a valid solution that the recommender system must obey. For example, a condominium buyer is likely to have some very strong constraints about location, price and amenities. As mentioned above, the tolerance for false positives is going to be low for high-risk items.

3.5.3 Churn

Recommender systems are used in domains with long-lived items like books, but they are also used in domains where the value or relevance of an item has a very short time span, such as news stories. A high churn domain is one in which items come and go rapidly.

In a high churn domain, a recommender system faces a continual stream of new items to be integrated into its knowledge sources. This greatly increases the sparsity of any kind of opinion data, as new items will necessarily have been seen by very few users. Items that have been around for some time may accumulate ratings, but by the time they do, they may no longer be relevant.

3.5.4 Interaction Style

In systems in which the user makes no special effort to interact with the recommender system, the system extracts the implicit expressed preferences from user behavior. For example, when visiting a Web site, a user leaves behind behavioral traces in Web server logs that can be used to make recommendations. Implicit inputs may include the specific items that the user is currently viewing, user transaction history, elapsed time, and shopping cart / purchasing behavior. Explicit inputs require that the user make the effort to formulate an opinion or a query or to add personal data to the system. There must be some perceived benefit for doing so that justifies the effort.

Implicit inputs are naturally noisy because they are inferred from user behavior. This type of interaction may be best suited for gathering simple rating knowledge, although some researchers have explored the extraction of preferences and even domain knowledge from implicit data [181, 230]. Explicit inputs may be more sparse if the burden of generating them causes users to do so relatively rarely.

3.5.5 Preference stability

User preferences can also have varying degrees of duration. For example, a person buying a digital camera would typically switch preferences after purchase, since they would be no longer interested once the purchase was complete. On the other hand, a person interested in comedy movies may wish to continue getting comedy recommendations for a long period of time. Also, preferences for some items may increase and wane naturally, for example, when one's favorite basketball team is in a big tournament, stories about it become highly preferred, but if they are knocked out or when the tournament is over, the user's preferences will change.

Stable preferences mean that opinion data collected in the past is still likely to be valid today. This makes it easier to maintain high-quality social knowledge sources. Unstable preferences mean that any data collected in the past may have to be discounted or discarded. The problem of preference instability can be ameliorated by collecting more data. If a user generates enough opinion data during a single session to adequately represent his or her current preferences, then there is no need to extrapolate from historical data and the issue of preference stability does not arise. This situation is found in Web personalization applications. Users generate a large number of clicks while browsing a Web site, enough implicit data to allow for recommendation using only the data from a single session.

3.5.6 Inscrutability

Certain applications (for example, high-risk ones) may require that the system is able to explain its recommendations, to answer questions like "why was this item recommended?" Such explanations enhance user confidence that a recommendation is appropriate [136] and increase the likelihood of recommendations being accepted.

Explaining recommendations is most straightforward when content knowledge sources are employed [135, 180]. Explaining a recommendation based on social knowledge has proved more challenging. Refer to Herlocker et al. [90] for an evaluation of some alternatives.

3.6 Mapping Knowledge Sources to Domain Properties

The characteristics of the domain of recommendation has a direct effect on the knowledge sources that the system can deploy. In this section, we look once more at the knowledge sources and discuss how the domain characteristics impact them. Table 3.1 summarizes the domain considerations and their impact on knowledge sources.

3.6.1 Individual

Individual knowledge is an essential requirement for a recommender system to produce personalized recommendations.

In collaborative recommendation, individual knowledge regarding the target user is matched against social knowledge drawn from the user population at large. The most straightforward version of the process is that these sources are of the same type, and all that is needed is a similarity metric by which individuals can be compared. In heterogenous domains, it might be difficult to transfer user's input on certain items for recommending other items. For example, it is not certain that two users who have similar taste about movies, would also like similar music. A solution to this problem can be taking into account the user context and only consider part of user profile that is related to the specific context of the recommendation. We proposed using semantic cues for contextual recommendation using such an approach in [172].

In domains with unstable preferences, user requirements are more likely to be needed since the historical data is unreliable. The query is the most fundamental form of input for requirements: the users state, in whatever form the system accepts, what it is they are looking for. Constraints and preferences allow the user to limit and to rank options. For example, a dog owner might have a strict constraint that any apartment he rents accept his pet. A parent with young children might have a preference to be close to parks and playgrounds. In high risk domains and domains which need explanation, it is usually necessary to have explicit requirements and constraints from the user. On the other hand, in domains with implicit interaction style or high churn user behavior and context can be considered as individual knowledge.

In many recommender systems more than one type of user input is used. For example, [137] uses users' priorities and constraints in a case based reasoning recommender.

3.6.2 Social Knowledge

Social knowledge enables the use of collaborative algorithms in which predictions about individuals are extrapolated from their peers opinions.

In heterogenous domains, social knowledge should be considered as a knowledge source since it is gathered by user's input and does not need extensive knowledge engineering. However, social knowledge is not sufficiently accurate and reliable for high risk domains or for domains which need explanation.

Social knowledge will tend to be sparse for high churn domains. When items come and go quickly, the odds are reduced that any given user will have a chance to rate any given item. As with other sparsity effects, the problem of churn can be ameliorated by having a large user population. Google News, for example, can take advantage of the site's large and active user base to implement collaborative recommendation even for the high-churn news domain [56].

Using social knowledge is appropriate for domains with implicit type of interaction since it is possible to mine users' behavior using machine learning and statistical techniques. In domains with unstable user preference, social knowledge can be misleading since the historical data is unreliable.

In the absence of social knowledge, individual knowledge especially in the form of ratings can also be combined with content knowledge in the form of item features to build a classic content-based recommender that uses supervised classification learning [147, 159].

3.6.3 Content and Domain Knowledge

As described before, the most basic kind of content knowledge is item features, the kind of data that would typically be available in a product database, such as numeric values like price or string values like destination airport. These features can typically be used as is in a recommender system, although implementers will often want to restrict the feature space. For example, the entire cast and crew list for a movie may be available as feature data but will contain many sparse features of little utility. Trimming this list to just the top billed actors, director, and screenwriter would probably be sufficient.

The quality of recommendations produced by a content-based or knowledge-based recommender will be entirely dependent on the quality of the content data on which its decisions are based. Indeed, the lack of reliable item features is often cited as a motivating factor for avoiding content-based recommendation. The cost involved in creating and maintaining a database of useful item features should not be underestimated, particularly for heterogeneous domains. In a domain like electronics products, for example, new technical innovations arrive regularly, requiring that the schema and the individual entries for each item be updated. In domains with high churn automatic approaches for feature extraction should be considered. If there are a large number of not-entirely-independent features extracted in a variety of ways, the system may be tolerant of noisy feature data. On the other hand, applications with high risk will need to pay special attention to having clean item features. Typically, manual review of feature data or manual labeling will be required. Using content knowledge makes the recommendations more explainable than the social knowledge. However, domains that need clear explanation for recommendation, should use domain knowledge as a reliable and explainable knowledge source.

Domain knowledge refers to more complex notions of content knowledge such as a feature ontology, domain constraints, or means-ends knowledge. A feature ontology relates features to each other so that similarity and difference between items can be more adequately assessed. Means-ends knowledge can help to determine what means/features are appropriate for which goals/ends that the user might have in mind. Knowledge-based recommenders usually need to get explicit user requirements in form of queries or constraints. Implicit interactions are hard to be matched against domain knowledge.

High risk domains should consider constraints of the domain. For example, a recommender for financial products [63] may know that certain investment instruments are only suitable for customers with certain characteristics – for example, a particular life insurance policy might not be available to persons over the age of 55.

Since domain knowledge usually needs to be developed by domain experts, it is costly and time consuming in heterogeneous domains to create and maintain domain knowledge. Moreover, it is not feasible to constantly update such knowledge in high churn domains.

Characteristic	Individual	Social	Content
Heterogeneous	May transfer poorly to unseen items	Appropriate	Difficult to engineer / maintain
High risk	Requirements and constraints usually needed	Not sufficiently accurate or reliable	Domain constraints needed
High churn	User behavior and context can be considered	Sparse data	Automatic feature extraction needed
Interaction style-Implicit	Detailed requirements not available	Appropriate	Difficult to match with domain knowledge
Unstable preferences	Historical data unreliable Historical data unreliable	/ user requirements needed	Appropriate
Inscrutability-Explanations needed	Requirements can be mapped to items	Explanations weak	Domain knowledge can be used

Table 3.1: Impact of recommendation domain on knowledge sources

3.7 Mapping Domains to Technologies

As we have shown above, the choice of domain and the characteristics of the application place certain constraints on the kinds of knowledge sources that a recommender system may deploy. In turn, the availability and quality of knowledge sources influences what recommendation technologies can be used.

Some basic considerations come to the fore in considering the recommendation domain. First, there are some domain types for which social knowledge seems not very useful, in particular, high risk domains and ones with high churn that do not have enough user inputs. In high churn domains, there may not be enough time for an item to build up a reputation among a large number of peer users before it is replaced with other items.

When there is large risk associated with a domain, most users are going to need a more convincing explanation of the appropriateness of a recommendation beyond simply that others liked it.

Similarly, if we look at the interaction, we can see that it is not always possible to gather every kind of knowledge type from every type of interaction. In systems with implicit inputs,

Factor		Collaborative	Content-Based	Knowledge-Based
Heterogeneous	Low	✓	✓	✓
	High	+	–	–
Risk	Low	✓	✓	✓
	High	–	–	+
Churn	Low	✓	✓	✓
	High	–	+	–
Interaction style	Implicit	+	+	–
	Explicit	✓	✓	+
Preferences	Stable	✓	✓	✓
	Unstable	–	✓	+
Scrutability	Required	–	✓	+
	Not Required	✓	✓	✓

Table 3.2: Domain factors and recommendation techniques. ✓: possible, +: recommended, –:not recommended

we do not gather any kind of direct requirements from the user (although it is sometimes possible to extract an implicit query from the user's activity with other applications, as done in the Watson system [30].)

Preference instability favors knowledge-based techniques since the historical data is unreliable. Learning over a user's prior interactions may turn out to be a hinderance rather than a help. However, in certain cases, such as Web personalization, users may provide enough input in a single session to form a useful profile that can be compared to others.

Table 3.2 shows the influence of the different domain factors on the choice of recommendation approach. In cases where the criteria do not help to reach a definitive conclusion, it is worth noting that the different technologies do have different implementation and maintenance costs. Collaborative recommendation is likely to be the least expensive to implement. It requires a database of user ratings, but it does not require clean, well-engineered item features, which is the minimum requirement for the other recommendation technologies. Knowledge-based technologies are going to be the most expensive approach requiring knowledge engineering and continuing maintenance. So, a developer might wish to start by implementing the least expensive solution compatible with the domain.

Another factor to consider is that with hybrid recommendation it is possible to combine techniques. For example, to deal with a heterogeneous environment with unstable preferences, a hybrid between content-based and collaborative recommendation may be desirable.

3.7.1 Algorithms

If a domain can be clearly characterized as appropriate for one recommendation technology or another, a natural next question is which algorithms are appropriate? A thorough treatment of this question is beyond the scope of this chapter. In the case of collaborative recommendation, however, it is possible to put forward some considerations.

In user-based collaborative recommendation, a subset of appropriate users are chosen based on their similarity to the active user , and a weighted aggregate of their ratings is used to generate predictions for the active user at run-time. Different implementations of collaborative filtering apply variations of the neighborhood-based prediction algorithms. Herlocker et al. [89] presents an empirical analysis of design choices in such algorithms and analyzes the variations of similarity metrics, weighting approaches, combination measures, and rating normalization.

Item-based collaborative filtering explores the relationship between items as a function of how users have rated them. The item-based version of *KNN* algorithm has been shown to scale better and produce more accurate recommendation than user-based for large item collections [188].

Memory-based nearest-neighbor algorithms have two important computational limits: scale and sparsity. The need to compare each user against every other (n^2 comparisons) makes these techniques impractical for large collections. Also, the need to directly compare item ratings means that in very sparse collections, users may have very few neighbors.

In some databases, overall sparsity may hide the fact that there are dense sub-regions of the item space. Exponential popularity curves may make it possible to employ memory-based techniques because it is possible to find agreement among people or items in the dense sub-region and use that agreement to recommend in the sparse space [91]. (Jester [74] does this directly by creating a highly dense region of jokes rated by all users).

Dimensionality reduction (by way of singular value decomposition, latent semantic analysis, or other techniques) is by now a standard approach for coping with sparsity in ratings databases [187, 233]. Various forms of compression and/or dimensionality reduction usually require extensive off-line computation, but as a result scale much better. The movie rating data released for Netflix prize which was also used for KDD cup competition in 2007 is an example of large, sparse data which motivated many research groups to develop new model-based algorithms [179, 116].

Other model-based collaborative algorithms include different machine learning techniques such as Bayesian networks [28] , and clustering [28, 207]. Bayesian networks are more practical for domains with high user preferences stability so that the user preference changes slowly with respect to the time needed to build the model.

3.8 Sample Recommendation Domains

Table 3.3 illustrates the application of these criteria in 10 different domains where recommendation applications exist. CF, CB, and KB stand for collaborative filtering, content-based, and Knowledge-based respectively. Not all combinations of the six criteria are represented, but we can see that the considerations given above are predictive. The last column of the table shows the algorithm suggested by our framework and the column before shows examples of existing real world systems or in literature.

In a nutshell, we can observe that high-risk domains generally lead to knowledge-based recommendation; scrutability is also a good predictor of this. Heterogeneous domains with implicit input are handled largely with collaborative recommendation.

3.9 Domain of This Thesis: Social Web

Social Web applications help users to socialize or interact with each other throughout the World Wide Web. These applications provide a social platform for users for different purposes. In some applications such as Facebook [1] and Myspace[2] the focus is on people. Such sites promote the person as focus of social interaction. In other applications, socializing is typified by a sort of "hobby focus". For example, Websites such as Flickr[3] and Kodak Gallery[4] provide a social platform for sharing photography and Last.Fm[5] for sharing music with like-minded people or friends. Other social applications might focus on more serious or professional proposes. For example, LinkedIn[6] focuses on professional networking, Citeulike[7] and Bibsonomy[8] provide a social platform for scientific community to share resources and find new articles of their interest. Wikipedia provides a collaborative multi-language open encyclopedia that allows users to share their knowledge in a very broad range of topics. Social bookmarking applications such as Delicious[9] allow users to store, access and share their bookmarks.

The important characteristic of social Web applications is the amount of user generated data. While in traditional recommenders systems, users associate ratings to existing products (items), in social Web, users have more possibilities to interact with the system. First, they can introduce new items (such as new bookmarks, or new photos) to the system which increases the heterogeneity and churn of the domain. Second, users can associate more information to the items than just ratings. This information, called "annotation" in this chapter,

[1]www.facebook.com
[2]http://www.myspace.com/
[3]http://www.flickr.com/
[4]www.kodakgallery.com/
[5]http://www.last.fm/
[6]http://www.linkedin.com/
[7]http://www.citeulike.org/
[8]www.bibsonomy.org/
[9]http://delicious.com/

Domain	Risk	Churn	Heterogeneous	Preference	Interaction Style	Scrutability	Examples	Our Suggestion
News	Low	High	Low	Stable	Implicit	Not required	Google News [56]: CF , YahooNews[21] and [145, 128]: CB	CB
E-commerce	Low	High	High	Stable/ Unstable	Implicit	Not required	Amazon, eBay: CF	CF
Web Page Recommender	Low	High	High	Stable/ Unstable	Implicit	Not required	[31, 121, 11]: CF, Hybrid	CF
Movie	Low	Low	Low	Stable	Implicit	Not required	Netflix [157, 202], Movielens[77]: CF	CF
Music	Low	Low	Low	Stable	Implicit	Not required	Pandora and [84, 95, 45]: CB, Hybrid	CB, CF
Financial-services, Life-insurance	High	Low	Low	Stable	Explicit	Required	Koba4MS[62] FSAdvisor[63] [205]: KB	KB
Software Engineering	Low	Low	Low	Stable	Explicit /Implicit	Required	[44] and [98]: Hybrid, CB	CB, KB
Tourism	High	Low	Low	Stable/ Unstable	Explicit	Required	Travel Recommender [177] [127]: CB,KB	KB
Job search Recruiting	High	Low	Low	Stable	Explicit	Required	CASPER[119] and [130]:CB,KB	KB
Real Estate	High	Low	Low	Stable	Explicit	Required	RentMe [35] FlatFinder[213] and [231]:KB	KB
Social Web	Low	High	High	Stable	Implicit	Not required	Facebook, Delicious, Last.fm: CF, Hybrid	CF, CB

Table 3.3: Sample domains for recommendation

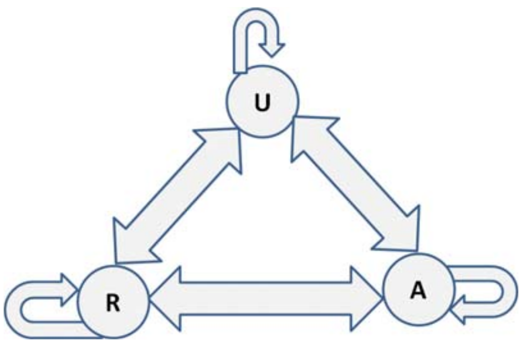


Figure 3.3: Possible relations in the graph of a social Web application

can have different forms such as tags, categories, reviews, and etc. Third, as the name suggests, the important feature of these kind of applications is their “social” characteristic. Users interact with social Web applications to be part of a community, to share information with others and to benefit from the collective knowledge of others for discovering new resources and new friends.

These properties of the social Web domain introduces new opportunities as well as new challenges to recommender systems. Users are presented not only with huge item space to select from but also annotations and other users. They also have the opportunity to contribute to the system; for example by assigning tags to resources. A recommender in social Web domain has to consider all different possible relations among users, items, and annotations in the social network.

It is useful to think of a social Web application as a graph with three types of nodes (users(U), annotations(A), and resources(R)). There are a variety of types of edges such a network can contain. Purely social edges connect two members of U: the “friends” relationship in Facebook, for example. Domain-specific links may connect members of R: for example, one Web page will often contain links to other pages. Even annotations might be linked to each other. For example, there may be an ontology in a tagging system by which one tag is related to another. Figure 3.3 shows the possible relations in a social graph.

The complexity of the navigational space in social applications makes accurate recommendations even more critical. While applications vary in the underlying resource and the form of the annotations, the model described above, nevertheless captures the essential elements of many social Web applications and can be used to characterize different recommendation tasks and modalities. For example, in traditional collaborative recommender applications such as in Netflix the user generates annotations that are typically numerical ratings and associates them to resources (movies). The recommendation task is to predict ratings for movies not yet rated by a target user.

Social tagging applications such as Delicious are another example of social annotation systems in which users annotate resources with tags rather than ratings. Social tagging sites routinely permit users to explore each others tags and resources creating rich navigational opportunities. Collaborative tagging gives rise to a variety of recommendation tasks: recommending resources, suggesting tags given a resource, or recommending other users with potentially similar interests.

Other forms of annotations can include reviews which are often incorporated into online retail stores such as Amazon or they can get a more structured form such as categories associated to pages in Wikipedia.

As the last row of table 3.3 shows, social Web domain is considered as high churn and heterogeneous domain. The heterogeneity and the fact that the items can be added by normal users at any time, impedes any knowledge-based possibility for recommendation in this domain. On the other hand, the high churn might make sparsity a problem for recommending some items. However, since the domain has the advantage of having many users interacting with the system very often, there can be possibility of collaborative recommendation for the dense part of the data. For the other parts, content-based approaches are appropriate.

The evolution of annotations from numerical ratings to other above mentioned forms break the assumptions of many traditional algorithms. These algorithms must either be adapted to the social Web context or new algorithms such as graph-based techniques need to be developed. In the next chapter, we discuss most effective tag recommendation techniques up to date and introduce novel approaches to improve them.

3.10 Conclusion

In this chapter, we presented a framework for matching domains to recommendation technologies with the goal of assisting researchers and developers in selection and application of these systems. Recommendation methods are usually classified into three main categories: collaborative, content-based, and knowledge-based. In contrast to previous taxonomies on recommender systems, our framework focuses on various characteristics of the domain. We identified six domain characteristics including heterogeneity, risk, churn, interaction style, preference stability, and inscrutability and and mapped these characteristics to the underlying knowledge sources and recommendation technologies. We presented guidelines for selecting among different kinds of recommendation technologies based on the problem characteristics. We provided examples from the real world applications and recommender system literature. Finally, we focused the social Web domain as it is the domain of this thesis; we presented the main differences of this domain with other ones and the important points that should be considered when designing recommenders in this domain.

3.11 Acknowledgements

The work presented in this chapter has been initiated by support of IBM Watson Research Center in summer 2007. An abbreviated version of the work appeared as a paper "Maryam Ramezani, Lawrence Bergman, Rich Thompson, Robin Burke and Bamshad Mobasher, *Selecting and Applying Recommendation Technologies*, Workshop on Recommendation and Collaboration at the Intelligent User Interfaces conference 2008." The work has been later continued and with additional contributions from Dr. Robin Burke, it has been published as a chapter in the recommender system hand book: " Robin Burke and Maryam Ramezani, *Matching Recommendation Technologies and Domains*, Recommender Systems Handbook: A Complete Guide for Research Scientists and Practitioners, Springer 2011."

Chapter 4

Improving Link Analysis for Tag Recommendation in Folksonomies

4.1 Introduction

Collaborative tagging systems such as Delicious[1], lastFm[2], and Bibsonomy[3] have emerged as powerful applications for Internet users. Tagging systems support users with several benefits. First they allow users to organize their own data with a level of freedom not possible in traditional taxonomic filing systems. Secondly they provide users with the means to openly share this information among friends and colleagues. Thirdly they also allow anyone to utilize the collective knowledge of others for discovering new topics, resources or perhaps even new friends.

While Folksonomies have many benefits, they also present several challenges. Most collaborative tagging applications permit unsupervised tagging; users are free to use any tag they wish to describe a resource. This unsupervised tagging can result in tag redundancy – in which several tags have the same meaning – or tag ambiguity – in which a single tag has many different meanings. Such inconsistencies can confound users as they attempt to utilize the folksonomy. It can be difficult for users to traverse the sheer volume of data. Moreover, noise in the data can impede the users experience. Data mining applications such as tag recommenders make it easier for the user to navigate the system.

Tag recommendation, the suggestion of tags during the annotation process reduces the user effort. By reducing the effort users are encouraged to tag more frequently, apply more tags to an individual resource, reuse common tags, and perhaps use tags the user had not previously considered. Moreover, user error is reduced by eliminating redundant tags caused

[1]http://delicious.com/

[2]http://www.last.fm/

[3]www.bibsonomy.org/

by capitalization inconsistencies, punctuation errors, misspellings and other discrepancies. The tag recommender can further promote a core tag vocabulary steering the user toward adopting certain tags while not imposing any strict rules. The tag recommender may even avoid ambiguous tags in favor of tags that offer greater information value. This may aid other users when navigating through the folksonomy to find interesting resources related to a tag which is more often used by other users. In order to develop a recommender applications in a social tagging system the first step is to create a model of the folksonomy that takes into account the information flow between users, resources and tags.

In this chapter we propose a weighted directed graph which models the informational channels of a folksonomy. We then apply PageRank to this model for tag recommendation. Our claim for a directed graph to model the folksonomy is based on the observation that the user navigation from one object (user, resource, or tag) to another object is not symmetric and by considering different weights on the edges of each direction we can better model the navigating from one node to the other.

4.2 Related Work

We provided an overview of existing approaches for tag recommendation in section 2.4.3. In this section, we look the most current state of the art of tag recommendation and present a comparison of the most successful tag recommendation approaches.

Hotho et al. suggest an approach for converting the folksonomy into an undirected tripartite graph and use an adaptation of PageRank (called Adapted PageRank or simply PageRank) to facilitate search and recommendation in folksonomies [29, 102, 101]. However, their representation based on an undirected graph has a shortcoming in that weights that flow in one direction of an edge will go back along the same edge in the next iteration of PageRank. To overcome this shortcoming, FolkRank was proposed. FolkRank has been one of the most successful tag recommendation algorithms in folksonomies to date. Rendle and Thieme [174] use pairwise interaction tensor factorization for tag recommendation. The algorithm explicitly models the pairwise interactions between users, items and tags. This approach was the most successful algorithm in the ECML/PKDD Discovery Challenge 2009 [60] for graph-based tag recommendation. Figure 4.1 shows the results from authors' experiments on comparison of their proposed approach with other popular tag recommendation algorithms. We can see in this figure that PageRank shows the worst results wile FolkRank and the suggested algorithm are comparable.

Our contributions in this chapter are two-fold. First, we demonstrate that with appropriate parameterizations, PageRank can outperform FolkRank in tag recommendation. Second, our extensive evaluation on three real world datasets reveal that PageRank generates better tag

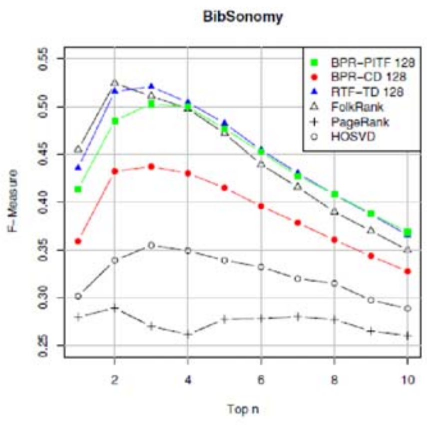


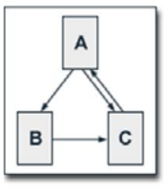
Figure 4.1: Comparion of different state of the art tag recommendation techniques taken from [174]

recommendations with a weighted directed graph model of the folksonomy than with an undirected graph model.

The rest of this chapter is organized as follows. In section 4.3, we provide a background on the PageRank algorithm and then we present the Adapted PageRank and FolkRank for folksonomies. Next, in section 4.3.3 we describe how to apply these algorithms for tag recommendation. Section 4.4 presents our proposed algorithm for creating a weighted directed graph from a folksonomy. Our experimental methodology, datasets and results are offered in section 4.5.2. Finally, we conclude the chapter with directions for future work.

4.3 Background on PageRank Algorithm

PageRank is a link analysis algorithm, introduced by Page and Brin [29] and it is the basic algorithm of the Google search engine. The underlying idea of PageRank is that pages that have many in-links from important pages are important themselves. PageRank uses probability distribution to represent the likelihood that a person randomly clicking on links will arrive at any particular page. The probability that the random surfer clicks on one link is given by the number of links on that page. For example, in figure 4.2 the probability of surfing from page B to page C is 1 while the probability of surfing from page A to page B is .5 since A has two out-links. The probability for the random surfer reaching one page is the sum of probabilities for the random surfer following links to this page. In our example, the probability of surfing to C is the sum of probability of surfing to B and probability



$$\begin{array}{c|ccc} & A & B & C \\ \hline A & 0 & 0 & 1 \\ B & .5 & 0 & 0 \\ C & .5 & 1 & 0 \end{array}$$

Figure 4.2: A simplified example for presentation of PageRank algorithm

of surfing to A. The PageRank algorithm considers that even an imaginary surfer who is randomly clicking on links might not click on an infinite number of links, but might get bored sometimes and jump to another page at random. At any step, the probability that the surfer does not follow the hyperlinks and instead jumps to a different page is γ. More formally, in PageRank, the authority of a page p, x_p, expresses the popularity of the page p and is defined based on the number of incoming links and on the authority of every page $q \in Q_p$ that connects to p with a forward link [19] and can be calculated as follows:

$$x_p = (1-\gamma) \sum_{q \in Q_p} \frac{x_q}{|H_q|} + \gamma \tag{4.1}$$

Where H_q is the set of all pages linked from q, and $|H_q|$ is the *outdegree* of q. The equation can be reformulated as the following linear system:

$$x = (1-\gamma)Wx + \gamma v \tag{4.2}$$

Here, the damping factor γ determines the influence of v, which is typically defined as $v = [1, \cdots, 1]^T$ but may be personalized with user preferences. The *transition matrix*, $W = \{w_{i,j}\}$ is defined as $w_{i,j} = 1/|H_j|$ if there exists a link from j to i and $w_{i,j} = 0$ otherwise. Every column must sum to either 1 or 0, making the matrix W column stochastic. The transition matrix for our simple example is shown in figure 4.2.

In order to compute personalized PageRank, v can be used to express user preferences by giving a higher weight to the components which represent the users preferred web pages. We call such vector, v, a "**preference vector**" and will use that for tag recommendation. In this case, the value of $\gamma \in [0,1]$ controls the influence of the preference vector. The preference vector allows the algorithm to be trained on a specific region of the graph.

4.3.1 Folksonomy-Adapted PageRank

Hotho et.al [100] introduced an adaptation to PageRank for folksonomies with the same basic notion that a resource which is tagged with important tags by important users becomes

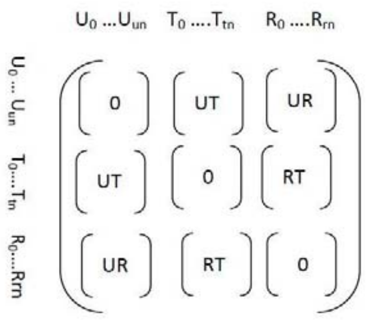


Figure 4.3: The folksonomy adjacency matrix

important itself. The same holds, symmetrically, for tags and users. Unlike the original PageRank that only deals with pages and links between them, the folksonomy Adapted PageRank considers three dimensions of users, resources, and tags. The folksonomy can be represented by a tripartite hypergraph in which the vertices are mutually reinforcing each other by spreading their weights. Hotho et. al use the projections introduced in section 2.3.2 and reduce the tripartite hypergraph to three bipartite graphs with regular edges, RT, UR, and UT, which reflect the relations between resources and tags, users and resources, and users and tags respectively. These three projections taken together produce a weighted undirected tripartite graph with set of nodes: $V = U \cup R \cup T$ and weighted edges $E = \{\{u,t\},\{r,t\},\{u,r\}|(u,t,r) \in A\}$, with each edge $\{u,t\}$ being weighted with $UT(u,t) = |r \in R : (u,t,r) \in A|$, each edge $\{t,r\}$ with $RT(r,t) = |u \in U : (u,t,r) \in A|$, and each edge $\{u,r\}$ being weighted with $UR(u,r) = |t \in T : (u,t,r) \in A|$. The adjacency matrix of this graph is shown in figure 4.3. The folksonomy transition matrix is built by normalizing each column of the matrix to 1. Such column-stochastic matrix is applied as W in equation 4.2 to find the rank of each element in folksonomy.

4.3.2 FolkRank

The problem with adapted PageRank is that it is based on an undirected graph while the origin of PageRank comes from the in-link and out-link concepts. Thus, in the Adapted PageRank algorithm, weights that flow in one direction of an edge will basically swash back along the same edge in the next iteration. Therefore the result is very similar to a ranking based on counting edge degrees [102]. Hotho et al. [102, 101] suggest FolkRank to solve this problem.

The FolkRank vector is taken as a difference between two computations of PageRank:

one with a preference vector and one without the preference vector. The preference vector (v in equation 4.2) is designed in a way that it gives higher weight to specific elements based the goal of ranking. For example, if the goal is to search for resources given a specific tag, the preference vector would give higher weight to that tag. More formally, if we consider x_0 to be the fixed point from equation 4.2 without preference vector and x_1 as the fixed point with preference vector p, then the FolkRank vector is defines as

$$x = x_1 - x_0 \tag{4.3}$$

Because a generic but popular item will receive a similar PageRank score in both models, its FolkRank will be reduced to 0 (or less, if its new score is less than the original). The inventors of FolkRank [102, 105] claim that x resulted from FolkRank provides valuable results on a large-scale real-world dataset while x_1, provides an unstructured mix of topic-relevant elements with elements having high edge degree.

However, our experimental results and analysis in the next sections show that with correct parametrization of the damping factor, adapted PageRank produces better results than FolkRank.

4.3.3 Graph-Based Tag Recommendation in Folksonomies

In traditional recommendation algorithms the input is often a user, u, and the output is a set of items, I. The user experience is improved if this set of items is relevant to the user's needs. Tag recommendation in folksonomies however differs in that the input is both a user, u, and a resource, r. The output remains a set of items, in this case a recommended set of tags, T_r. An algorithm for tag recommendation in folksonomies therefore requires a means to include both user and resource information in the process so that the recommendation set includes tags that are relevant to the resource and also represent the user's tagging practice.

In order to generate tag recommendations the preference vector is biased towards the query user and resource [105]. These elements are given a substantial weight in the preference vector where all other elements have uniformly small weights. Similar to [105] each user, tag and resource gets a preference weight of 1 while the target user and resource get a preference weight of $1+|U|$ and $1+|R|$, resp.

The value of γ controls the impact of the preference vector. The higher it gets, the more the algorithm is biased toward the target user and resource. However, in previous tag recommendation experiments, authors [105, 174] have only tested the algorithms by fixing the value of $\gamma = .3$ without any specific reasoning, which indeed provides the best results for FolkRank but not for the PageRank algorithm. In this work, we will show that with increasing the value of γ, FolkRank will not do as well and PageRank can provide much better results.

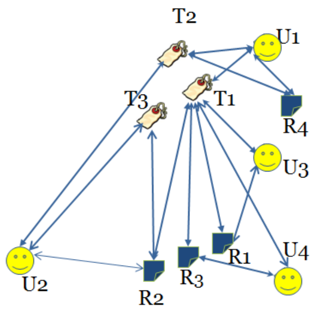


Figure 4.4: An arbitrary example of folksonomy

4.4 A weighted Directed Graph Model for Folksonomies

The problem with the weighting schema of Adapted PageRank is that the weight of the undirected edges does not accurately reflect the flow of information across the folksonomy. We extend this graph model to consider the expectation of hypothetical user navigating from one node to another. Consider r representing a non-popular resource and t representing a popular tag associated to r in a folksonomy. For instance, in figure 4.4, an arbitrary example is given where T_1 is a popular tag while R_3 is a non-popular resource. In the current weighting schema of Adapted PageRank, the weight between r and t is defined by the number of users who associate them together represented with RT(r,t):

$$RT(r,t) = |u \in U : (u,t,r) \in A| \tag{4.4}$$

Our weighting schema suggests that the weight from r to t should be higher than the weight from t to r since t is a popular tag and a user is more likely to navigate from a non-popular resource to a popular tag than navigating from a popular tag to a non-popular resource.

Our proposed weighting approach is inspired by the weighted PageRank algorithm [1], where the importance of pages are considered for weighting the graph. The weighted PageRank algorithm in [1] assigns larger rank values to more important (popular) pages instead of dividing the rank value of a page evenly among its out-link pages. Each out-link page gets a value proportional to its popularity (its number of in-links and out-links).

We take a similar approach and our goal is to assign different weights to each direction of the link based on the popularity of the in-link element. Since we face a folksonomy graph, we have three different types of vertices(U,R,T) and each edge connects two types of the vertices. The popularity of the elements is defined differently for each type of edge and is based on the number of connected elements of the third type of vertex. To clarify, consider an edge between resource r and tag t. Popularity of t is defined as the sum of all users who have used the tag t (sum of the in-links to tag t from the user nodes). The weight from r to t is defined as multiplicative factor of popularity and $RT(r,t)$. Similarly, the popularity of r will be the total of number of users who have annotated resource r in their profile. In our arbitrary example, the number of users who have tag T_1 in their profile is 3 while number of users who have resource R_3 is 1. So the weight of the link from R_3 to T_1 is three times higher than the weight from T_1 to R_3. Equations 4.5 and 4.6 show the formal definitions of edge weights from r to t and from t to r respectively.

$$weight(r \rightarrow t) = RT(r,t) \times \sum_{r' \in R} RT(r',t) \tag{4.5}$$

$$weight(t \rightarrow r) = RT(r,t) \times \sum_{t' \in T} RT(r,t') \tag{4.6}$$

The weight of an edge between tag t and user u is defined as follows. Popularity of tag t is the number of all resources associated with t and the popularity of user u is the sum of all resources tagged by the user u. Following equations show the formal definitions for the edge weights.

$$weight(u \rightarrow t) = UT(u,t) \times \sum_{u' \in U} UT(u',t) \tag{4.7}$$

$$weight(t \rightarrow u) = UT(u,t) \times \sum_{t' \in T} UT(u,t') \tag{4.8}$$

Weights of UR edges are defined similarly. Popularity of user u is defined as the number of all tags the user has applied and popularity of r will be the sum of number of tags assigned to r. Thus, the weights are defined as following:

$$weight(u \rightarrow r) = UR(u,r) \times \sum_{u' \in U} UR(u',r) \tag{4.9}$$

$$weight(r \rightarrow u) = UR(u,r) \times \sum_{r' \in R} UR(u,r') \tag{4.10}$$

In our arbitrary example, user 1 has associated tags T1 and T2 to resource R4. Thus, $UT(U1,T1) = 1$ and $UR(R4,U1) = 2$, considering our weighting schema, the weight from T1 to U1 is 1 while the weight from U1 to T1 is equal to 4 since T1 is associated to resources R1, R2,R3 and R4.

With the proposed weighting schema, we have a weighted directed graph and we can apply the PageRank algorithm. In the following section we use the suggested graph model for tag recommendation and compare the results with FolkRank and PageRank with undirected graph.

4.5 Experimental Evaluation

Our tag recommendation algorithm uses the PageRank algorithm with the directed graph described above. We also experimentally analyze the effect of changing the parameter γ in equation 4.2 for the three approaches: PageRank with undirected graph, FolkRank, and PageRank with directed graph. As described before, for all experiments, we bias the preference vector by setting the weight of $1+|U|$ and $1+|R|$ for the target user and resource respectively, while other users, tags and resources get equal weight of 1.

First, we describe the methods used to gather and pre-process data for the experiments and provide details of our datasets. Next, we discuss our testing methodology and briefly explain the common metrics recall and precision and then detail the results of our experiments.

4.5.1 Datasets

We have chosen three datasets for our experiments: Delicious, Bibsonomy and Citeulike. In order to reduce noise and focus on the denser portion of the dataset a P-core was taken such that each user, resource and tag appear in at least p posts as in [16, 105]. A post is defined as as a user, resource and all tags applied by that user to that resource and an annotation is defined as a user, a resource and one tag associated to that resource by that user.

Delicious

Delicious[4] is a popular Web site in which users annotate URLs. On 10/19/2008, 198 of the most popular tags were taken from the user interface. For each of these tags the 2,000 most recent annotations including the contributors of the annotations were collected. This resulted in 99,864 distinct usernames. For each user, the “Network” and “Fans” were explored recursively collecting additional usernames. A user’s Network consists of the other users that the user has explicitly chosen to watch. Conversely a Fan is another user that has explicitly chosen to watch the user. For each user the social network was explored recursively resulting in a total of 524,790 usernames.

From 10/20/2008 to 12/15/2008 the complete profiles of all users were collected. Each user profile consisted of a collection of posts including the resource, tags and date of the original bookmark. The top 100 most prolific users were visually inspected; twelve were removed from the data because their post count was many orders of magnitude larger than other users and were suspected to be Web-bots. Due to memory and time constraints, 10% of the user profiles were randomly selected. A p-core of 20 was taken from this dataset for experiments.

[4]www.delicious.com/

Bibsonomy

This dataset was provided by Bibsonomy for use in the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD) 2009 Challenge. Bibsonomy was originally launched as a social annotation system allowing users to organize and share scholarly references. It has since expanded its scope allowing users to annotate URLs. The data includes all public bookmarks and publication posts of Bibsonomy until 2009-01-01. A 5-core was taken to reduce noise and increase density.

Citeulike

Citeulike is a popular online tool used by researchers to manage and discover scholarly references. They make their dataset freely available to download. On 2/17/2009 the most recent snapshot was downloaded. The data contains anonymous user ids and posts for each user including resources, the date and time of the posting and the tags applied to the resource. A *P*-core of 5 was calculated.

Folksonomy	**Delicious**	**Citeulike**	**Bibsonomy**
Users	7,665	2,051	402
Resources	15,612	5,376	2,014
Tags	5,746	3,343	1,755
Posts	720,788	42,278	15,760
Annotations	2,762,235	105,873	53,554
Average # of tags in each post	3.82	2.5	3.39
Average # of post per user	94.04	20.61	39.2
Average # of posts per resource	46.1	7.86	7.82

Table 4.1: Datasets

Table 4.1 summarizes the base statistics for each dataset. From these fundamental numbers we derive a few simple statistics which help to characterize the "density" of the post-processed datasets. The average number of tags in a Delicious post is 3.82, and the average number of posts per user is 94.04. For Citeulike, the averages are 2.50 tags per post and 20.61 posts per user; for Bibsonomy, 3.39 and 39.2. We therefore note that Delicious has by far the most tagging data per user, followed by Bibsonomy, and Citeulike last. In terms of posts per resource, we again see that Delicious has far and away the highest average with 46.1; Citeulike and Bibsonomy are nearly identical in this respect (7.86 vs. 7.82, respectively).

An important distinction between the datasets is their focus. Users in Delicious are able to tag any URL available on the Web. As such an individual's interests are often varied encompassing many topics. In Citeulike, however, researchers tag scholarly publications and their tagging is often focused in their area of expertise. Due to its dual-nature, we expect Bibsonomy users to display a somewhat mixed approach.

We have adopted the test methodology as described in [105]. In this approach, called *LeavePostOut*, a single post is randomly removed from each user's profile. The training set is then comprised of all of the remaining data, while the test set contains one test case per user. Each test case consists of a user, u, a resource, r, and all the tags the user has applied to that resource. These tags, T_h, are analogous to the holdout set commonly used in Information Retrieval. The tag recommendation algorithms accept the user-resource pair and return an ordered set of recommended tags, T_r. From the holdout set and recommendation set utility metrics were calculated. For each evaluation metric the average value is calculated across all test cases.

For evaluation we use the common recall and precision measures as is common in Information Retrieval. Recall is a measure of completeness and is defined as:

$$r = |T_h \cap T_r| / |T_h| \tag{4.11}$$

Where T_h is the hold-out set(test set) tags which represent the correct answer and T_r is the set of recommended tags. Recall measures the percentage of items in the recommendation set that appear in the holdout set. Precision measures the exactness of the recommendation algorithm and is defined as:

$$p = |T_h \cap T_r| / |T_r| \tag{4.12}$$

which measures the percentage of items in the holdout set that appear in the recommendation set. In order to be able to compare the performance of the algorithms, we use F-measure defined as follows:

$$\textit{F-Measure} = \frac{2 \times \textit{Precision} \times \textit{Recall}}{\textit{Precision} + \textit{Recall}} \tag{4.13}$$

4.5.2 Experimental Results

Here we present our experimental results with tuning of value of γ and changing the number of recommended tags. We have recorded the values of precision and recall with changing the number of recommended tags from 1 to 10 and also changing the value of γ between .05 to 1. The Adapted PageRank results are shown with abbreviation of "APR" in the graphs. Figures 4.5 through 4.7 show the effect of changing the value of γ on precision and recall for the three data sets. The charts show the precision and recall values for recommendation set of 10 tags (Recall at 10 and Precision at 10) when changing the value of γ.

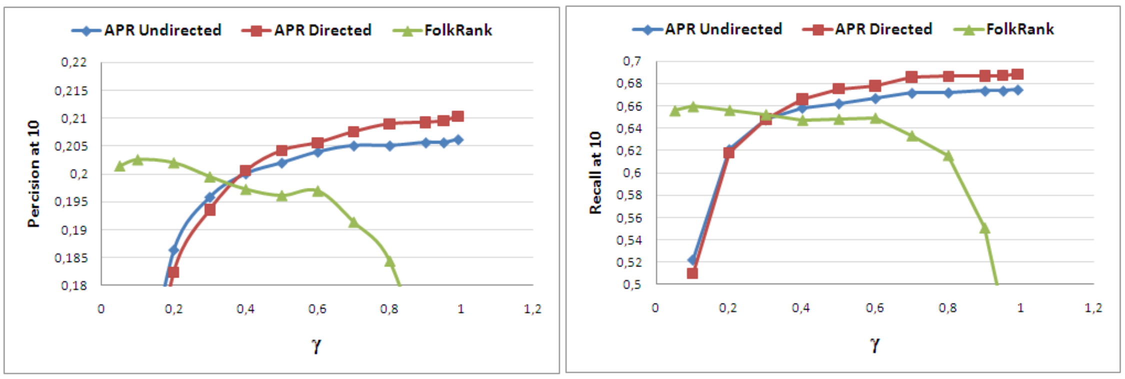


Figure 4.5: The effect of changing γ on precision and recall for a recommendation set of 10 tags in Bibsonomy data set

Increasing the value of γ indicates a greater emphasis on the preference vector and our results show that increasing γ has a consistently positive effect on tag recommendation using the Adapted PageRank algorithm. Even though the results for $\gamma = .99$ still show an improvement, setting the value of $\gamma = 1$ will drastically reduce the recommendation precision to nearly zero. This is expected, as $\gamma = 1$ in equation 4.2 means that there is no learning involved and the final resulting vector is equal to the preference vector.

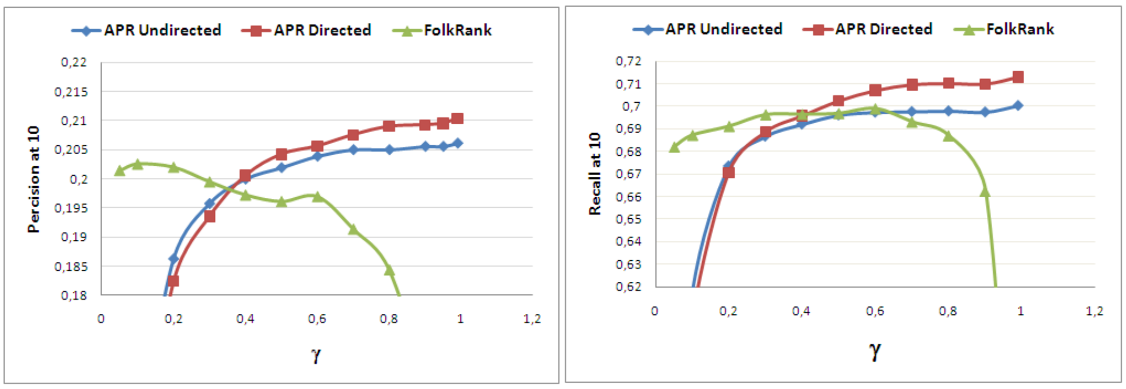


Figure 4.6: The effect of changing γ on precision and recall for a recommendation set of 10 tags in Citeulike data set

Our results show that for FolkRank, on the other hand, by increasing the value of γ, the performance decreases. To explain this we should note that FolkRank is basically the difference of the resulting weights of the adapted PageRank with and without preference vector. As the value of γ increases the result is more and more focused on the preference vector which results in higher weights for the nodes that are connected to the target nodes. The differential approach in FolkRank is supposed to compute a topic-specific ranking of the elements in a folksonomy. However, the fact that the weights from the PageRank without preference (x_0) are subtracted from the one with preference vector (x_1) creates negative values in the resulting vector for any node which has a higher value in x_1. This is useful for small values of γ to remove the high weighted popular nodes, however, as the value of γ increases x_1 is more and more representative of the actual ranks considering the specific resource and tag. By subtracting x_0 from x_1 we end up with an ad-hoc unrealistic weight vector, since we might omit many high weighted nodes in x_1 only because they have occurred with a higher weight in x_0.

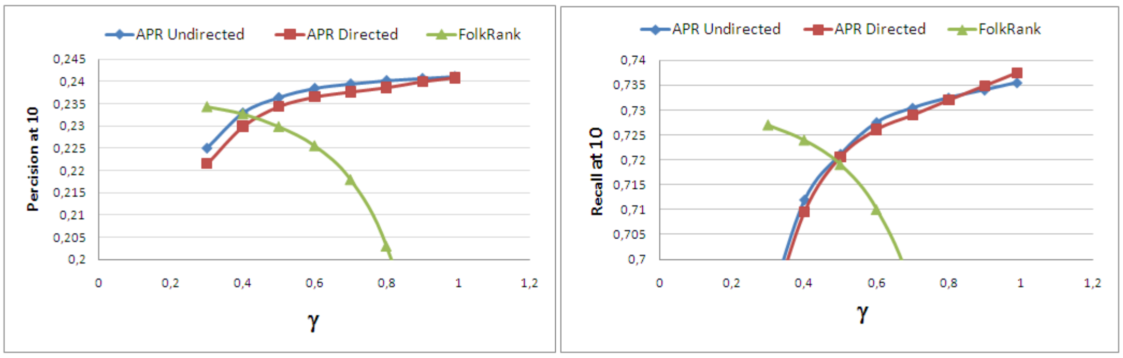


Figure 4.7: The effect of changing γ on precision and recall for a recommendation set of 10 tags in Delicious data set

The above charts show the results when the number of recommended tags is fixed to 10. We want to examine if this results are valid when we change the number of recommended tags. Figure 4.8 shows the results for different values of precision and recall when recommending from 1 to 10 tags. In this experiment, we keep the value of γ constant and set it to the value that results in best performance for each technique. From figure 4.6 we can observe that this value is .3 for FolkRank and .99 for Adapted PageRank. In this figure we can observe that the adapted PageRank using the directed graph outperforms the undirected version and the FolkRank for all values of precision and recall. Figure 4.9 shows similar results comparing the F-measure for different approaches. This figure confirms that the performance of the adapted pagerank is better than FolkRank, and that the directed graph model outperforms the undirected one for recommendation of any number of tags.[5] Note that the rightmost point of this graph is the one which is presented in figure 4.6. Although in this figure it seems that all points converge to the same value, our significance tests show that the differences are significant even for that point.

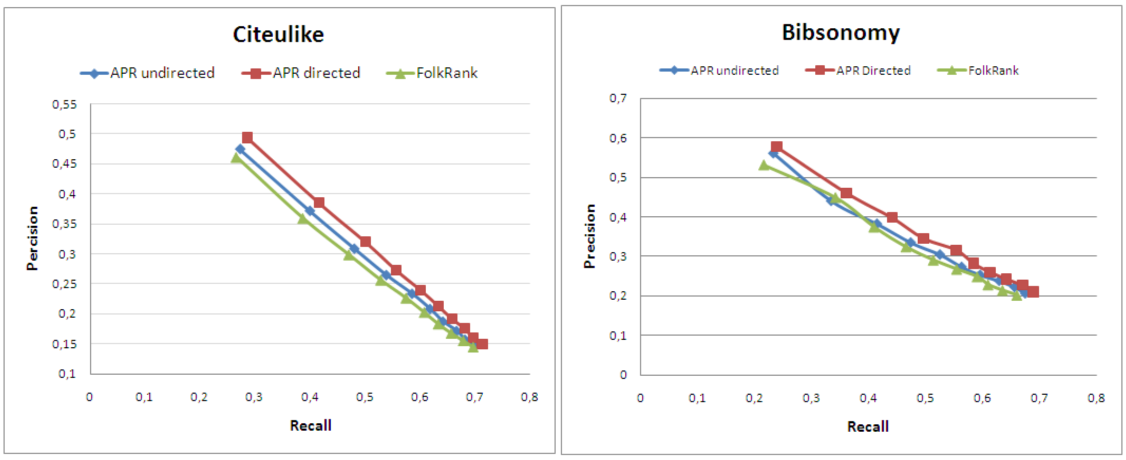


Figure 4.8: Comparison of undirected APR, directed APR for $\gamma = .99$ and FolkRank for $\gamma = .3$

4.5.3 Discussion

The results from different data sets show that increasing the value of γ results in increasing the recommendation accuracy of the PageRank algorithm and it has the opposite effect for the FolkRank. We can see that performance of FolkRank drops when the value of γ increases. The comparison of directed and undirected graphs show that directed graph produces better or similar results across different datasets. In Bibsonomy and Citeulike the results of the directed graph are considerably better for certain values of γ.

We performed significance test to determine if the differences between observed values from each approach in different data sets are significant. We used pair sample t-test and compared the mean of precision and recall resulted from the constant γ which produces the best results for each technique. The results from the significance test show that the differences between the values from the directed graph is significantly better than the the undirected

[5]The difference between the F-measure values in figure 4.9 and 4.1 is because of the difference in the data set used in our experiments. Our experiments were performed before the release of the new data set by Bibsonomy.

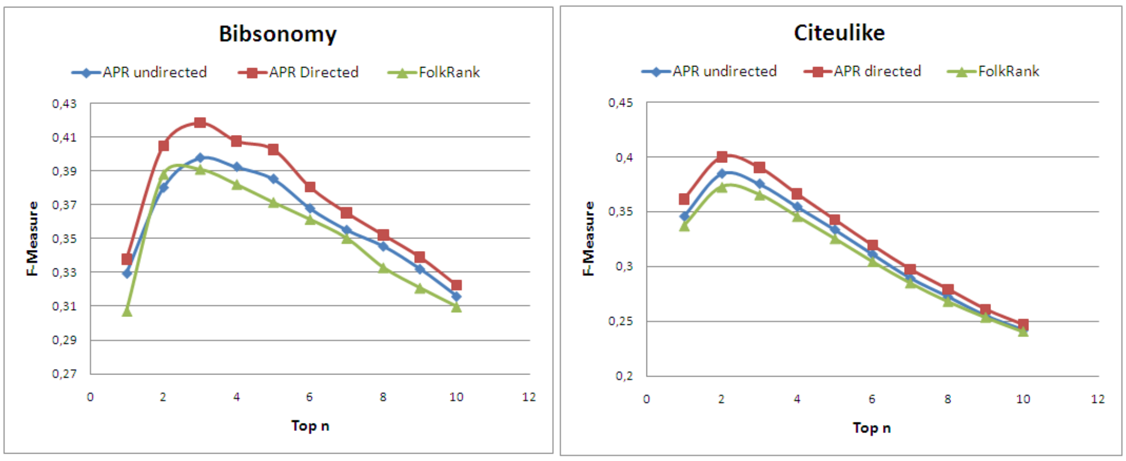


Figure 4.9: Comparison of F-measure for undirected APR,directed APR for $\gamma = .99$ and FolkRank for $\gamma = .3$

one for Bibsonomy and Citeulike datasets with $p < .001$. However, in Delicious dataset the differences are not significant which means that we can not reject the null hypothesis that the two approaches (directed and undirected graph models) produce similar results. This difference is likely because of existence of much broader concepts in Delicious and deserves further investigation. The significance tests show that in all datasets the Adapted PageRank significantly outperforms FolkRank.

4.6 Conclusion and Future Work

In this chapter we suggested to model the folksonomy as a weighted directed graph which can capture the informational channels of a folksonomy. We then applied PageRank to this model for tag recommendation. Our extensive evaluation on three real world datasets reveal that with appropriate parameterization, Adapted PageRank can outperform FolkRank in tag recommendation. In addition, we have shown that with modeling the folksonomy as a directed weighted graph, we can get additional improvement. Future work can improve the weighting schemas to better model the probability of navigation in the folksonomy and apply the graph models for search and resource recommendation. Moreover, efficiency and scalability are major drawbacks of this algorithm which deserve more attention. In the current setting, since the preference vector is dependent on the target resource and user, all the iterative calculation must be done in online recommendation phase which depending on the size of the folksonomy can be quite time consuming. Developing new algorithms that can help calculate the personalized PageRank with an incremental approach -so that the complete iteration calculations are not necessary at the run time- can be a valuable future work which can improve personalized search and recommendation not only in the social tagging environment but also in other Web applications.

4.7 Acknowledgment

The work presented in this chapter was supported by DePaul University Summer Research Grants. Thanks to Jonathan Gemmell, Thomas Schimoler, and Bamshad Mobasher for their help and contribution to this work.

Chapter 5

Using Recommender Systems to Support Continuous Ontology Development

5.1 Introduction

Ontologies are known in computer science as consensual models of domains of discourse, usually implemented as formal definitions of the relevant conceptual entities [87]. There are two broad schools of thought on how ontologies are created: the first views ontology development akin to software development as a - by and large - one off effort that happens separate from and before ontology usage [154, 201]. The second view is that ontologies are created and used at the same time, that ontologies are continuously developed throughout their use. The second view is exemplified by the ontology maturing model [25, 24] and by the ontologies that are developed in the course of the usage of a semantic wiki [114].

Machine learning, data mining and text mining methods to support ontology development have so far focused on the first schools of thought; have focused on creating an initial ontology from large sets of text or data that is refined in a manual process before it is then used. For example, recently, many researchers have tried to apply data mining techniques to create ontologies from folksonomies [93, 13].

In our work, however, we focus on using machine learning techniques to support continuous ontology development. We suggest a new kind of semantic tagging system which starts with a seed ontology – created by domain experts – that assists users of the system with recommendations on how to further develop and mature that ontology. In particular, we focus on one important decision: given the current state of the ontology, the concepts already present and the sub/super concept relations between them - where should a given new concept be placed? Which concept(s) should become the super concept(s) of the new concept? Our work presents a supporting tool for user-driven ontology evolution management proposed by Stojanovic et al. [200].

We investigate this question on the basis of applications that use ontologies to aid in the structuring and retrieval of information resources (as opposed to for example the use of ontologies in an expert system). These applications associate concepts of the ontology with information resources, e.g. a concept "Computer Science Scholar" is associated to a text about Alan Turing. Such system can use the background knowledge about the concept to include Alan Turing page in responses to queries like "important British scholars".

The rest of this chapter is organized as follows. Section 5.2 provides a brief background on ontologies as well as a summary of the previous related work. In section 5.3 we present several application scenarios of our work. Section 5.4 presents our proposed algorithms for the recommendation of super-concepts. In section 5.5, we describe the methodology, the datasets and the results from the evaluation before section 5.7 concludes the paper.

5.2 Related Work

According to one of the most cited definitions of the Semantic Web literature, an ontology is a formal, explicit specification of a shared conceptualization [79]. Guarino clarifies Grubers definition by adding that the AI usage of the term refers to "an engineering artifact, constituted by a specific vocabulary used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary words" [80].

An ontology is engineered by members of a domain by explicating a reality as a set of agreed terms and logically founded constraints on their use [139]. The ontology construction either uses a bottom-up methodology which starts with some descriptions of the domain and then obtains a classification or a top-down methodology which starts with an abstract view of the domain itself, which is given a priori [55].

One of the most important components of ontologies are concept hierarchies, which model the information of the domain in terms of concepts and subsumption relationships between them [103]. A concept hierarchy defines a sequence of mappings from a set of lower-level concepts to their higher-level correspondences. Such mappings may organize the set of concepts in partial order, such as in the shape of a tree (a hierarchy, a taxonomy), a lattice, a directed acyclic graph, etc., although they are still called "hierarchies" for convenience [83].

John Sowa [199] classifies ontologies into three kinds. A *formal ontology* is a conceptualization in which categories are described by axioms and definitions. They are stated in logic or in some computer-oriented language that could be automatically translated to logic. In contrast, in *prototype-based ontologies*, categories are distinguished by typical instances or prototypes rather than by axioms and definitions in logic. Categories are formed by collecting instances rather than describing the set of all possible instances in an intensional way, and the most typical members are selected for description. The third kind of ontology is *terminological ontology* which is specified by subtype-supertype relations. The concepts are described by concept labels or synonyms rather than prototypical instances [20]. A well known example for a terminological ontology is WordNet [140]. Figure 5.1 illustrates different ontology paradigms for a toy example in the food domain taken from [20]. In this work, we focus on terminological ontologies specifically on subtype-supertype relations among concepts.

Ontologies can be learnt from various sources, structured and unstructured documents, databases or more recently from folksonomies. Maedche and Staab [126] present an ontology learning framework that extends typical ontology engineering environments by using semi-automatic ontology construction tools. The framework includes ontology import, extraction, pruning, refinement, and evaluation. The authors also present a survey on ontology learning approaches from text, dictionaries, and legacy ontologies. We presented a detailed related work on the area of emergent ontologies from folksonomies in section 2.4.5. Here, we review the related work specifically on machine learning techniques for ontology learning.

Ontology learning techniques can be divided to two groups: constructing ontologies from scratch and extending existent ontologies. The former includes various clustering techniques (unsupervised learning) and the latter can be viewed as a classification task (supervised learning) [20].

Different approaches of clustering has been applied for ontology learning from text. Hierarchical clustering for ontology learning allocates sets of terms in a hierarchy that can be transformed directly into a prototype-based ontology. A distance or similarity measure on terms is defined, and the terms with higher similarity than a specified threshold are merged into one cluster. An overview of clustering methods for obtaining ontologies from different sources is presented in [48].

In addition to hierarchical clustering, other data mining and statistical techniques have been used for ontology learning. Maedche and Staab [125] present a general architecture for discovering conceptual structures and engineering ontologies. Based on this architecture, they apply association rule mining to learn ontologies from textual data. Chen and Li [46] use spectral graph partitioning to construct concept hierarchy from web pages. Sanderson and Craft [184] use subsumption which is basically based on co-occurrence of terms to automatically create concept hierarchy from text. Aussenac-Gilles et al. [12] use natural language processing techniques to analysis a corpus for creation of ontologies. A semi-automatic construction of ontologies using text is presented in [43]. The proposed method compares the relative frequency of terms in the text with their typical, expected use and identifies concepts and relations. For a comprehensive overview on ontology learning from

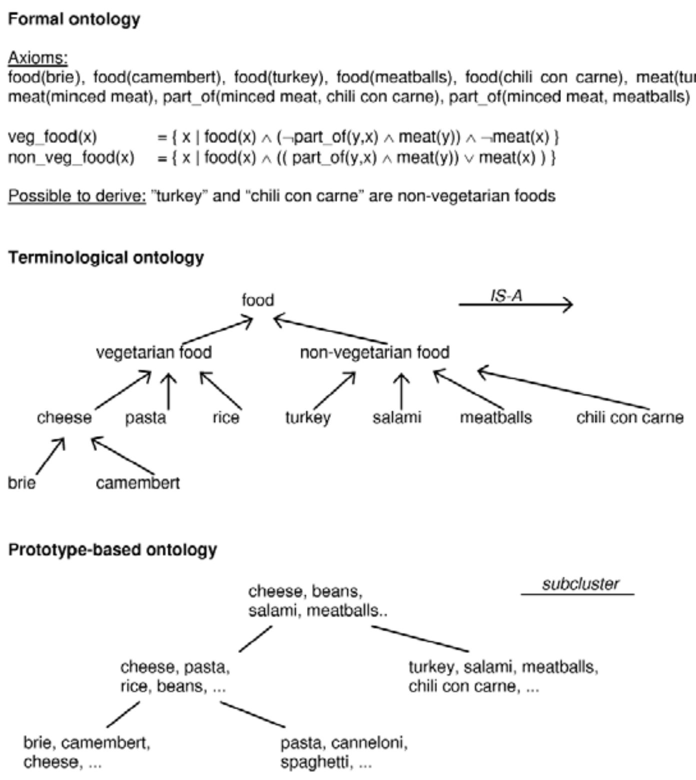

Formal ontology
Axioms:
food(brie), food(camembert), food(turkey), food(meatballs), food(chili con carne), meat(turkey), meat(minced meat), part_of(minced meat, chili con carne), part_of(minced meat, meatballs)
veg_food(x) = { x | food(x) ∧ (¬part_of(y,x) ∧ meat(y)) ∧ ¬meat(x) }
non_veg_food(x) = { x | food(x) ∧ ((part_of(y,x) ∧ meat(y)) ∨ meat(x)) }
Possible to derive: "turkey" and "chili con carne" are non-vegetarian foods
Terminological ontology
food
IS-A
vegetarian food
non-vegetarian food
cheese
pasta
rice
turkey
salami
meatballs
chili con carne
brie
camembert
Prototype-based ontology
cheese, beans, salami, meatballs..
subcluster
cheese, pasta, rice, beans, ...
turkey, salami, meatballs, chili con carne, ...
brie, camembert, cheese, ...
pasta, canneloni, spaghetti, ...

Figure 5.1: Formal vs. terminological vs. prototype-based food ontology [20]

text, methods, evaluation and applications, refer to [158].

On the other hand, extension of an existing ontology can be viewed as a classification task. Features of the existing data can be used as a training set to design a classifier for new instances. Our work in this chapter is an extension of such approach in Web 2.0 environment. Thus, we elaborate more on the previous works in this specific area.

Alfonseca and Manandhar [8] propose enrichment of WordNet with new concepts learnt from general purpose texts. For each new concept, the algorithm performs a top-down search along the ontology, and stops at the synset that is most similar to the new concept. The ontology is traversed top-down from the root until an appropriate position is found. The largest problem with this approach is the general nature of top-level concepts that leads to taking the wrong path in the beginning of the process. Similar work is presented by Witschel [220] in which decision trees are used for extending lexical taxonomies. In this work, co-occurrence statistics is used to discover the relations among concepts and the semantic descriptions are propagated iteratively upwards to the root.

Fleischman and Hovy [65] focus on one specific category of named entities, *persons*, and describe a method for automatically classifying person instances into eight sub-categories, e.g politician, entertainer, etc . A supervised learning method is suggested that considers the local context surrounding the entity as well as more global semantic information derived from topic signatures and WordNet. An unsupervised algorithm is presented by Widdows [218] for placing unknown words into Wordnet taxonomy. The algorithm first finds semantic neighbors of the unknown word by applying latent semantic analysis with part-of-speech information. Then, the unknown word is placed in the part of the taxonomy where these neighbors are most concentrated, using a class-labeling algorithm.

Cimiano et al. [49] propose the learning of taxonomic relations as a classification task. They suggest using Hearst-patterns [86] for learning taxonomic relations by considering various and heterogeneous forms of evidence including large text corpus, World Wide Web, and Wordnet.

Our approach is similar to the classification tasks described above with several differences. First, most the abovementioned techniques are appropriate for a taxonomic relationship where each concept has only one super-concept while our approach does not have this limitation. Our algorithms can propose several potential super-concepts ordered by the associated confidence. Second, our approach suggests a recommendation algorithm to support end users for continuous ontology development collaboratively [25, 24] , i.e. where anybody can add a new element to the ontology, and refine or modify existing ones in a work-integrated way. That means the ontology is continuously evolving and gradually built up from social tagging activities and it is not derived once at a specific time. Our work provides a supporting tool for such ontology building by helping users with recommendation of taxonomic relations. We continue this chapter by first providing several application scenarios of our work and then description of the algorithms.

5.3 Application Scenarios

We present several application scenarios for this work. The examples include industry applications as well as social Web applications. All these systems are "Web 2.0" style semantic applications; they enable users to change and develop the ontology (or concept hierarchy) during their use of the system. Our work assists users in this task by utilizing recommender system algorithms. The algorithms suggest potential super-concepts for new concepts introduced to the system.

5.3.1 Floyd

Floyd is a new product developed at SAP for supporting knowledge intensive enterprise processes such as product design, product management, market research, etc. Market researchers and support experts use Floyd to analyze and solve technical problems that their customers communicate them. The system supports several functionalities including semantic annotations, semantic search, semantic navigation, and semantic suggestions for relevant terms. Our work suggests an approach for improving the semantic suggestions functionality. Semantic suggestions are used when a term is entered into the system that does not exist in the current ontology. Users enter new terms into the system in two basic situations: (1) annotations (assigning tags to documents) (2) search (when the user tries to retrieve information about a term).

Semantic suggestions have to take into account the user input and the context. Examples of semantic relations include polysemy (ambiguous user input should be disambiguated), synonymy, and abstraction (super-concepts of a particular concept or term). The Floyd system is initially deployed with a semantic network developed by domain experts based on the existing company vocabulary.

5.3.2 SOBOLEO

SOBOLEO [229] is a system for web-based collaborative engineering of SKOS ontologies and annotation of web resources. Users collaboratively create a taxonomy, use it to annotate web resources and as background knowledge during search. Figure 5.2 shows a screenshot of the system. SOBOLEO consists of 4 main parts:

- **Search:** A search engine that searches through the annotated web resources using the taxonomy as background knowledge.
- **Browse:** An interface to browse through the taxonomy and the annotated resources.
- **Annotate:** An annotation interface to add bookmarks to the index. Users can use concepts from the taxonomy or arbitrary tags to annotate web resources.
- **Edit:** A collaborative real time editor for the taxonomy - all users can observe the changes others make on the taxonomy in real time. The taxonomy is collaboratively edited by everyone using the system.

There are several other examples of similar social semantic bookmarking application [26].

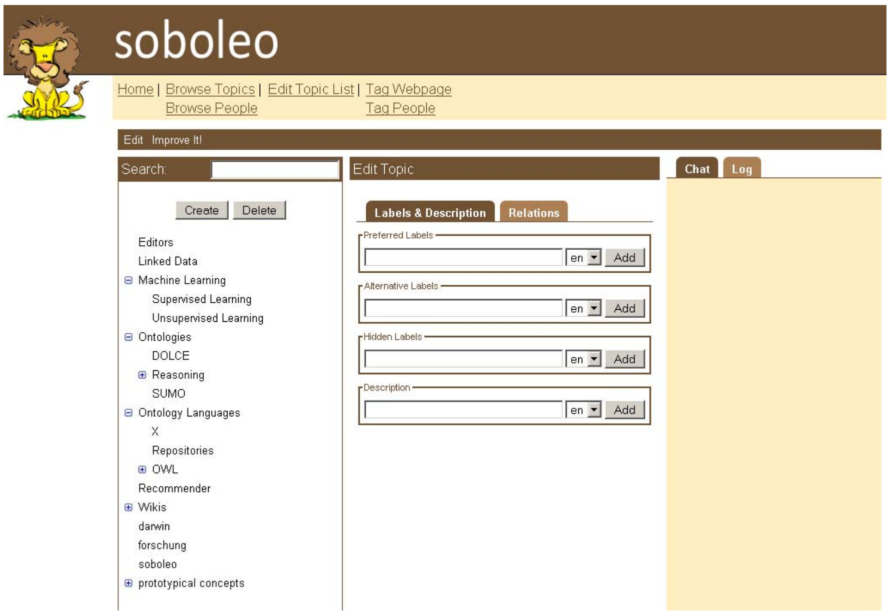


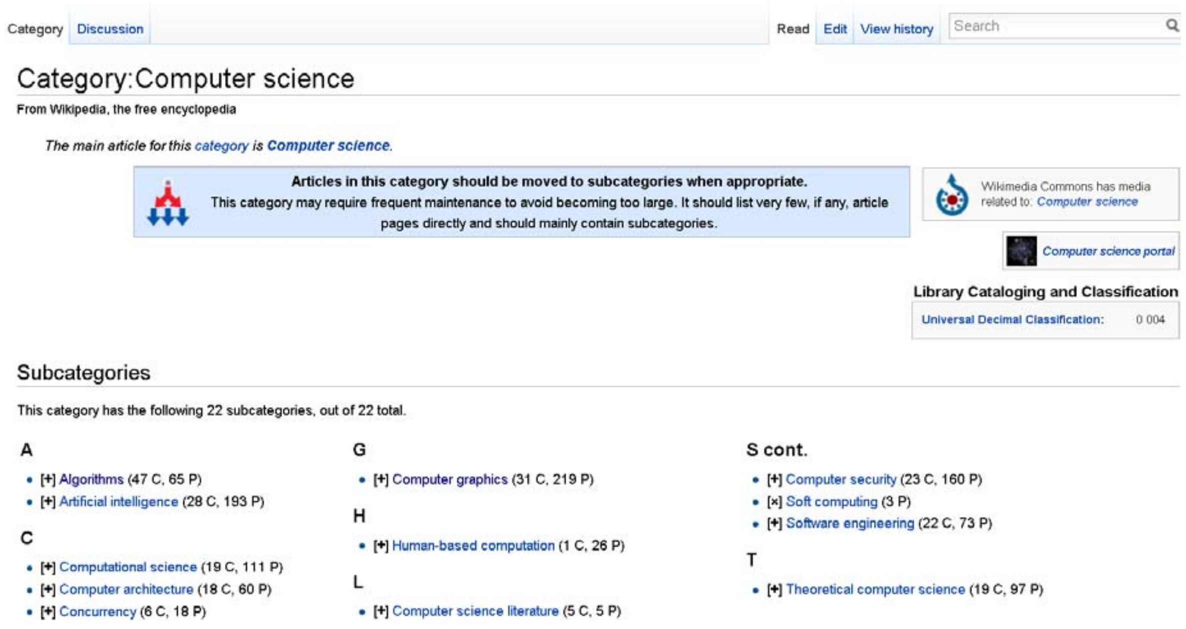
Figure 5.2: Screenshot of SOBOLEO system

5.3.3 Wikipedia

Wikipedia is the world's largest collaboratively edited source of encyclopedic knowledge. It is based on the MediaWiki[1] software. The idea of Wikipedia is to allow everyone to edit and extend the encyclopedic content. Wikipedia uses a hierarchy of categories to classify articles according to their content and they are mainly used to assist browsing. Category links are always displayed at the bottom of an article to help the readers for navigating through the system. Figure 5.3 shows screenshot of the sub-categories of the category of "computer science". We can see that as in many other broad categories, Wikipedia suggests to move the articles to appropriate sub-categories to facilitate search and navigation. Users can have a better search experience if the category hierarchy is complete and articles are associated to the relevant sub-categories.

We can view categories as akin to concepts and support the creation and placement of new categories by proposing candidate super-categories.

[1]http://www.mediawiki.org


Category Discussion Read Edit View history Search

Category:Computer science

From Wikipedia, the free encyclopedia

The main article for this category is Computer science.

Articles in this category should be moved to subcategories when appropriate.
This category may require frequent maintenance to avoid becoming too large. It should list very few, if any, article pages directly and should mainly contain subcategories.

Wikimedia Commons has media related to: *Computer science*

Computer science portal

Library Cataloging and Classification

Universal Decimal Classification: 0 004

Subcategories

This category has the following 22 subcategories, out of 22 total.

A
- [+] Algorithms (47 C, 65 P)
- [+] Artificial intelligence (28 C, 193 P)

C
- [+] Computational science (19 C, 111 P)
- [+] Computer architecture (18 C, 60 P)
- [+] Concurrency (6 C, 18 P)

G
- [+] Computer graphics (31 C, 219 P)

H
- [+] Human-based computation (1 C, 26 P)

L
- [+] Computer science literature (5 C, 5 P)

S cont.
- [+] Computer security (23 C, 160 P)
- [×] Soft computing (3 P)
- [+] Software engineering (22 C, 73 P)

T
- [+] Theoretical computer science (19 C, 97 P)

Figure 5.3: Screenshot of category hierarchy in Wikipedia; The sub-categories of “Computer science” and suggestion of Wikipedia to move the articles to appropriate subcategories

5.4 Algorithms

We propose three algorithms for the recommendation of super-concepts for a new concept. First, we presume that a seed hierarchy is not available or if there is one, it is not populated enough to be used for learning purpose. Second, we assume to have an existing concept hierarchy and our goal is to assist users to place a new concept in the right place in the hierarchy. Finally, we present a hybrid algorithm which combines the first two algorithms.

5.4.1 Algorithm 1: Recommendation of Super-Concepts Without An Existing Seed Hierarchy

We propose algorithm 1 for cases where little data and (almost) no existing hierarchy is present (for example for cases where a system is just starting to be used). For these cases we propose to use a string-based approach inspired by [96]; an approach focusing on compound terms. There are different approaches to find the similarity of two concepts only based on string-based attributes. The common approaches included edit-distance, token-based, and hybrid distance functions. Refer to [148] for overview and comparison of different approaches. These approaches are pretty well-established in the literature and our goal is not to contribute to string-based similarity functions but just use them as a first step for our recommendation. We use two of the string-based approaches: Soft-TFIDF and Jaccard similarity. We selected Jaccard similarity for its simplicity and Soft-TFIDF since it has been shown to be the most successful algorithm for string name matching in comparison with other approaches[52].

Jaccard Simialrity

Two strings $S1$ and $S2$ can be considered as multi-sets (or bags) of words (or tokens). The Jaccard similarity between the word sets S and T is simply defined as:

$$J(S1,S2) = \frac{S1 \cap S2}{S1 \cup S2} \tag{5.1}$$

Soft-TFIDF

TFIDF is a measure widely used in information retrieval and can be also used to compute string similarity between bags of words in a corpus. In this case, the TFIDF value represents the importance of each feature w (e.g. word) for an entity S belonging to the set E of entities:

$$\mathit{tf(w,S)} = \frac{n_{w,S}}{\sum_{w'} n_{w',S}}, \mathit{idf(w))} = log \frac{|E|}{|S \in E|w \in S|} \tag{5.2}$$

$$\mathit{TFIDF(w,S)} = \mathit{tf(w,S)} \times \mathit{idf(w)} \tag{5.3}$$

Where $n_{w,S}$ is the number of times w appears in S. The TFIDF similarity between S1 and S2 is defined as the COS similarity between the TFIDF vector representations of the S1 and S2.

Soft-TFIDF is a hybrid similarity measure introduced by [52] which is designed to take advantage of the good performance of TFIDF, without automatically discarding words which are not strictly identical. Soft-TFIDF combines TFIDF with Jaro-Winkler distance [219] which is a measure based on the number and order of the common characters between two strings. We have used the available java implementation provided by [52]. For more details of the algorithm, please refer to [52].

Super-concept Recommendation

We first compute the similarity between the name of the new concept and the names of concepts already known to the system (the candidate super-concepts) as described above. The super-concept recommendation is then based on the observation that the head of a compound is often a super-concept of the compound itself [96]. Therefore, we can observe that sub-concepts generally consist of more terms than their super-concepts, and – if they are formed out of super-concepts by adding so-called modifiers – usually also have significant string overlap with their super-concepts. For example "Computer Science" is a sub-concept of "science".

Thus, we define recommendation confidence as $W(S,T) = Sim(S,T)$ if S has fewer terms than T and 0 otherwise. More formally:

$$W(S,T) = \begin{cases} Sim(S,T) & \text{If } \sum_{w \in S} n_{w,S} < \sum_{w' \in T} n_{w',T} \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

Where Sim(S,T) is the string-based similarity and $n_{w,S}$ is the number of times w appears in S.

To find the super concepts for a new concept C_t, we find $W(C_i, C_t)$ where $C_i \in E$ represents all existing concepts in the existing concept-hierarchy E. Concepts with the highest W can be recommended as possible super concept for the new concept. We can recommend a list of top n concepts with highest W where n is a pre-fixed number or we can use a threshold value and only recommend concepts with W higher than the threshold.

Other imaginable algorithms for this task could build on data distribution properties such as entropy or co-occurrence. However, we do not consider those approaches here because our main goal in this chapter is to learn from an existing hierarchy. We only suggest algorithm 1 for cases that there is not enough data to learn from, and in such cases, statistical approaches are also not a practical approach. Our main contribution in this chapter will be presented in the next section, in algorithm 2, in which we assume to have a seed hierarchy to start with.

5.4.2 Algorithm 2: Recommendation of Super-Concepts by Learning From an Existing Concept Hierarchy

We propose algorithm 2 for cases where there already exists a sizeable hierarchy. The abstract idea for the proposed algorithm is to place a new concept at a location in the hierarchy where the most similar concepts to the new concept are located. We first present a simple approach to measure the degree of sub-super relationship between concepts in a concept hierarchy. Our algorithm tries to predict this value for the new introduced concepts.

Degree of Sub-Super Relationship in a Concept Hierarchy

There are various approaches to calculate the semantic similarity between concepts in a concept hierarchy. The simplest approach is probably the edge-counting which goes back

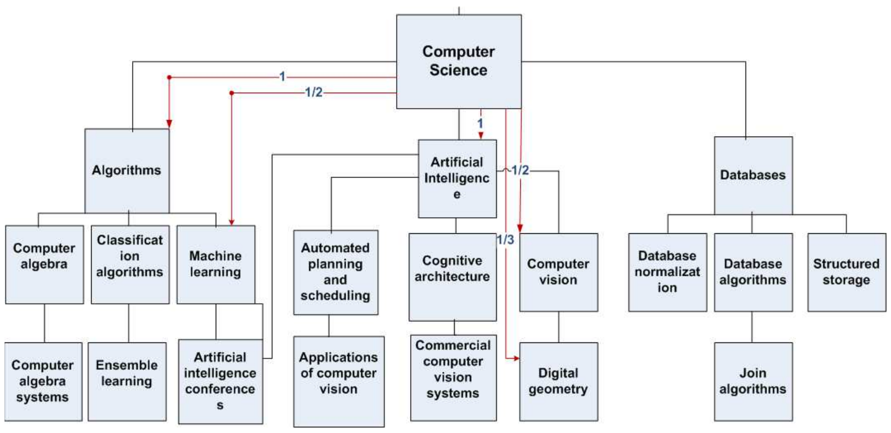


Figure 5.4: An example from a part of concept Hierarchy from Wikipedia. We define "SSA" as the reverse of shortest path distance between the super-sub concept.

to Quillians semantic memory model [53]. Later, more factors including depths of concept nodes, the density of the hierarchy, link type, and link strength were considered to enhance the path length approach [223, 225].

In this work, we are interested in measuring the degree of sub-super relationship between two concepts rather than the semantic similarity of them since our goal is to find the super-concepts for a particular new concept. We consider the reverse of the shortest path distance between two concepts, starting from the sub-concept and allowing only upward edges to be used to arrive at the super-concept. We call this "super-sub affinity" ("SSA"). To clarify how we find SSA, consider the concept hierarchy in figure 5.4. In this example the SSA(Computer Science,Algorithms)=1, SSA(Computer Science, Computer Vision)=1/2 and SSA(Algorithms, Structured Storage)=0. Note that SSA is not a symmetric relation, distinguishing it from common semantic similarity measures. In fact, the definition of SSA would entail "if SSA(A,B)≠0 then SSA(B,A)=0". We define SSA(A,A)=1. Thus, we can store all SSA values of a concept hierarchy in an $n \times m$ matrix where n is the number of concepts which have at least one sub-concept, m is the total number of concepts in the hierarchy, and the matrix diagonal is always 1. We will use this matrix in our recommendation algorithm described in section 5.4 for discovering super-concepts. The transpose of this matrix can be used for suggesting sub-concepts using the same algorithm. However, in this work, we focus only on recommending super-concepts.

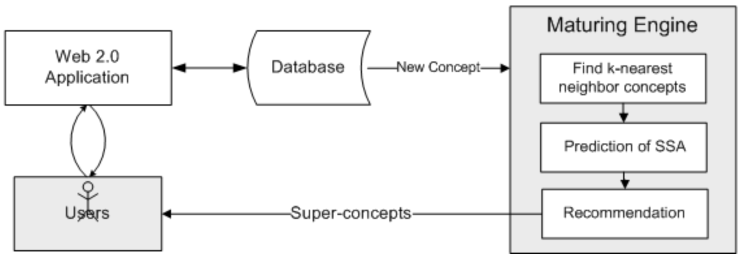


Figure 5.5: Recommender Architecture

General Architecture

Figure 5.5 shows the general architecture of our system. Generally, we assume that users interact with a Web 2.0 application. While they do so, the database gets updated with new concepts that they enter, for example, tags in a social tagging system or categories in a Wiki. The new concepts are then introduced to the maturing engine, which recommends potential super-concepts to the user once it has enough information about the new concept. If the user accepts the recommendation, the new sub-super relationship will be added to the concept hierarchy. The maturing engine executes three main steps to compute the potential super-concepts: Similarity computation, SSA prediction, and recommendation.

Similarity Computation

To find the similarity between a new concept and existing concepts, we consider measures that use similarities in the concept names as well as measures that use contextual cues.

For **string-based similarity** we use the same approaches already described in section 5.4.1, Jaccard and Soft-TFIDF. We define $Sim_s(C_i, C_t)$ as the string-based similarity between two concepts C_t and C_i. We also define the set of k most similar concepts to the target concept C_t and call this set N_s(string-based neighborhood).

Context-based similarity aim at using the context that the new concept has been used in to find similar concepts. Context has been defined by Dey [57] as any information that can be used to characterize the situation of an entity. In a social tagging system, for example, the context of a new tag entered into the system can be determined by the related resources, links between the resources, users who enter the tag, time, language and geographical information. In this work, we use the resources associated to a concept as a feature set to determine the context of the concept. We represent each concept, C, as a vector over the set of resources, where each weight, $w(r_i)$, in each dimension corresponds to the importance of a particular resource, r_i.

$$\vec{C} = \langle w(r_1), w(r_2) ... w(r_{|R|}) \rangle \tag{5.5}$$

In calculating the vector weights, a variety of measures can be used. The weights may be binary, merely showing that one or more users have associated that concept to the resource, or it may be finer grained using the number of users that have associated that concept to the resource. With either weighting approach, a similarity measure between two vectors can be calculated by using several techniques such as the Jaccard similarity coefficient or Cosine similarity [211]. Cosine similarity is a popular measure defined as

$$\mathit{Cosine(C1,C2)} = \frac{C1.C2}{||C1||\,||C2||} \tag{5.6}$$

In this work, we use binary weighting for representing concepts as a vector of pages and Cosine similarity to find similar concepts. Thus, the similarity between each concept C_i and the new target concept C_t is defined as $sim_c(C_t,C_i) = Cosine(C_t,C_i)$. Using this similarity measure, we find the set of k most similar concepts to the target concept C_t and we call this set N_c(context-based neighbors).

We define a **hybrid similarity measure** by combining the string-based and contextual-based similarity measures. For that purpose, we use a linear combination of the similarity values found in each approach.

$$Sim_h(C_i,C_t) = \alpha Sim_s(C_i,C_t) + (1-\alpha)Sim_c(C_i,C_t) \tag{5.7}$$

where $Sim_h(C_i,C_t)$ is the hybrid similarity value, and α is a combination parameter specifying the weight of string-based approach in the combined measure. If $\alpha = 1$, then $Sim_h(C_i,C_t) = Sim_s(C_i,C_t)$, in other words the neighbors are calculated only based on string-similarity. On the other hand, if $\alpha = 0$, then only the contextual information is used for finding similar concepts. We choose the proper value of alpha by performing sensitivity analysis in our experimental section.

Prediction Computation

Based on the Super-Sub Affinity and the similarity measures defined above, we can now predict the degree of sub-super relationship (SSA) between the new concept and every other concept in the hierarchy. Our proposed algorithm is inspired by the popular weighted sum approach for item-based collaborative filtering [188].

We predict the SSA values for a new concept by averaging over the *actual* SSA values between the neighbors of the new concept and the existing concepts. Formally, we predict the SSA between the target concept C_t and all other concepts C_i in the hierarchy as follows.

$$SSA_p(C_i,C_t) = \frac{\sum_{C_n \in N} SSA(C_i,C_n) * sim(C_t,C_n)}{\sum_{C_n \in N} sim(C_t,C_n)} \tag{5.8}$$

where $SSA_p(C_i, C_t)$ stands for the predicted SSA value for the pair (C_i, C_t), $SSA(C_i, C_n)$ stands for the actual SSA for (C_i, C_n), and $sim(C_t, C_n)$ is the similarity value between the target concept and neighbor concept which can be either string-based (Sim_s), contextual (Sim_c) or the hybrid (Sim_h) similarity. Thus, N can be either N_s, N_c or N_h as described in section 5.4.2. Basically, SSA_p is predicted based on the location of the existing concepts that are similar to C_t; it becomes large when many of C_t's neighbors are close to the current candidate concept C_i in terms of SSA. Hence, the best candidates for becoming a super-concept of C_t are those C_i for which $SSA_p(C_i, C_t)$ is maximal. The weighted sum is scaled by the sum of the similarity terms to make sure the prediction is within the predefined range. In this work we have defined the direct sub-super affinity as 1. Thus, the nearer the prediction of $SSA(C_i, C_t)$ to 1, the more probable that C_i is a good super-concept of C_t.

Recommendation

Once the SSA values for all existing concepts and the new concept are calculated, the concept(s) with the highest SSA can be recommended as super-concept(s) for the new concept. We can recommend a list of top n concepts with highest SSA prediction or we can use a threshold value and only recommend concepts with predicted SSA higher than the threshold. The threshold value (between 0 and 1) represents the "confidence" of the algorithm in recommendations. If there are no similar concepts found in step 1 or the predicted SSA values are lower than the threshold, the system does not make a recommendation which might mean that the new concept should be added as a new independent concept in top of the hierarchy or that the system is not able to find the right place for the new concept.

5.4.3 Algorithm 3: Hybrid Recommendation

As third algorithm we propose a combination of the two algorithms presented above based on the well known idea of hybrid recommender systems that combine two or more recommendation techniques to gain better performance with fewer of the drawbacks of any individual one [32]. We combine algorithm 1 and algorithm 2 using a linear combination approach:

$$W_h(C_i, C_t) = \beta W(C_i, C_t) + (1 - \beta) SSA_p(C_i, C_t) \tag{5.9}$$

where β is a combination parameter specifying the weight of algorithm 1 in the combined measure. If $\beta = 1$, then $W_h(C_i, C_t) = W(C_i, C_t)$, in other words we have only algorithm 1 based on string-similarity and if $\beta = 0$, then only SSA from algorithm 2 is used. Finding the appropriate value for β is not a trivial task, and is usually highly dependent on the characteristics of the data.

5.5 Evaluation

In this section, we describe the data sets we used for evaluation, our evaluation methodology, and our experimental metrics.

5.5.1 Data Sets

Wikipedia

To test our algorithms we need a Web 2.0 application where users can easily add new concepts and create semantic relations. We decided to use Wikipedia which is the most suitable Web 2.0 application at hand. Hepp [195] theoretically proves Wikipedia as a reliable and large living ontology. We treat the categories of Wikipedia as concepts and the existing relationships between "Sub-categories" as the seed concept hierarchy. Each category in Wikipedia has several associated pages, which we use as a context vector for the category as described in section 5.4.2. Thus, each category is represented as a binary vector over the set of pages. The weight of each page r_i for category C_j is 1 if page r_i is associated to category C_j and 0 otherwise.

For running our experiments, we focused on a small part of the English Wikipedia. We started from the category "Computer Science" as the root concept and extracted the subcategories by traversing with breadth first search through the category hierarchy. Our final data set has over 80,000 categories. For our experiments we created three smaller data sets to compare how the size and properties of the seed concept hierarchy impact the results. Table 5.1 shows the properties of each data set from Wikipedia.

Data Properties	**Large**	**Medium**	**Small**
Number of Concepts	24,024	9931	3016
Number of Pages	209,076	107,41	47,523
Average depth of the hierarchy	9	6	3

Table 5.1: Properties of the the Wikipedia category data for different sizes

Floyd

As second data set we utilize data from the use of the prototypical Floyd system at SAP. This data set provides a seed ontology in the sense that 241 known hierarchical relations between term concepts are defined in the semantic network.

Unfortunately, the user base for the Floyd system is still too small to yield any significant context data and the semantic network which is to be used as the seed concept hierarchy is still under development. So, only algorithm 1 could be evaluated on this data. It is still interesting to investigate this data set, especially because of its differences with the Wikipedia data since it is used within the marketing domain. The system contains mainly named entities – names of products, competitors, and also terms describing the market addressed by the company. These terms are quite different from Wikipedia categories and are from a rather specialized vocabulary.

5.5.2 Evaluation Methodology

We divided each data set into a training set and a test set. Since we were interested to know how the density of the seed ontology affects the results, we introduced a variable that determines what percentage of data is used as training and test sets; we call this variable x. A value of x = 20% would indicate 80% of the data was used as training set and 20% of the data was used as test set. We remove all information of test cases from the data set to evaluate the performance of the algorithm. For each experiment, we calculate the SSA values before and after removing the test cases. If the test case has sub-concept and super-concept, after removing the test case, its sub-concepts will be directly connected to its super-concepts and the algorithm has to intelligently discover its original place. Since algorithm 1 does not need training, we expect no change in performance of the algorithm with different test/train ratio and we can use leave one out methodology for experiments of algorithm 1.

5.5.3 Experimental Metrics

For the evaluation we adopt the common recall and precision measures from the Information Retrieval community. Recall is a common metric for evaluating the utility of recommendation algorithms. It measures the percentage of items in the holdout set that appear in the recommendation set. Recall is a measure of completeness and is defined as:$recall = |C_h \cap C_r|/|C_h|$ where C_h is the set of holdout concepts and C_r is the set of recommended concepts. Precision measures the percentage of items in the recommendation set that appear in the holdout set. Precision measures the exactness of the recommendation algorithm and is defined as: $precision = |C_h \cap C_r|/|C_r|$

In order to be able to compare the performance of the algorithms, we use the F-measure defined as following:

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5.10}$$

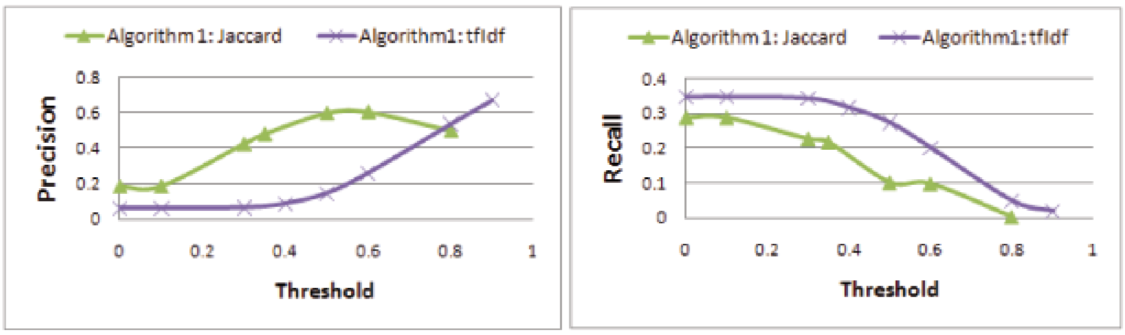


Figure 5.6: Result of algorithm 1 for Wikipedia data set

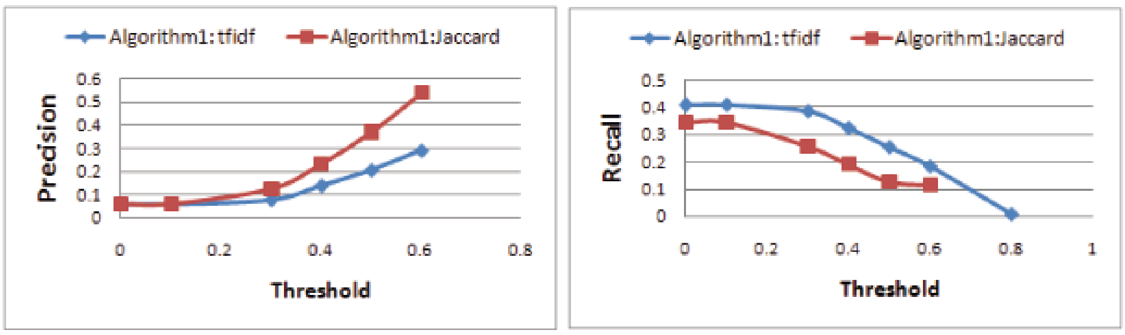


Figure 5.7: Result of algorithm 1 for Floyd data set

5.6 Experimental Results

In this section we present our experimental results of applying the proposed recommendation algorithms to the data sets described above. In all experiments, we change the threshold value from 0 to 0.95 and record the values of precision and recall. As the threshold value increases, we expect precision to increase and recall to decrease since we recommend less items with higher confidence. In addition, to get a better view of performance of the algorithms, we use the notion of recall at N. The idea is to establish a window of size N at the top of the recommendation list and find recall for different number of recommendations. As the number of recommendations increases, we expect to get higher recall.

5.6.1 Experimental Results from Algorithm 1

We applied the string-based techniques discussed in section 5.4.1 for recommendation of super-concepts for both data sets. The results of those experiments are shown in figure 5.6 and 5.7. The charts show the precision and recall as we increase the threshold value. The maximum threshold value for each case is the maximum possible string-similarity in the data set which is not necessarily 1. We can observe a similar trend for both data sets which can confirm that our evaluation with Wikipedia data set is applicable to an actual real data set. From these charts, we can also observe that while Jaccard similarity seems to deliver better precision, TFIDF similarity provides better recall for almost all threshold values.

5.6.2 Experimental Results from Algorithm 2

We applied the recommendation algorithm described in section 5.4.2 to the Wikipedia data set. Similar to algorithm 1, we change the threshold value from 0 to .95 and record the values of precision and recall. Before we compare the performance of different similarity measures, we run several experiments to find the sensibility of different parameters including the data size, the neighborhood size, and the combination factor for the hybrid similarity.

Effect of the neighborhood size

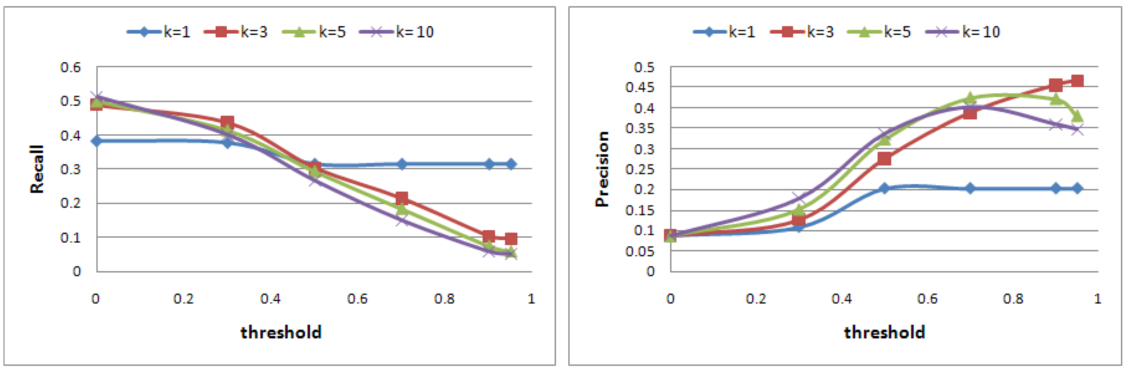


Figure 5.8: Precision and Recall for different threshold values for different neighborhood values (k)

To determine the influence of neighborhood size, we performed an experiment where we varied the number of neighbors to be used and computed precision and recall. The results of this experiment for the medium data set is shown in figure 5.8. From this figure, we can observe that there is a trade-off between better precision or better recall depending on the neighborhood size. We select $k = 3$ for our neighborhood size which gives better precision for high threshold values assuming that precision is more important for users. In a real scenario, depending on the goal of the system, the neighborhood size can be tuned for the optimum performance which can be high precision, high recall or a middle point with acceptable values for both.

Note that k=1 can be considered as a baseline algorithm since it simply finds the most similar concept to the new concept and recommends its super-concepts.

Effect of data set size

To determine the effect of the size of the data in our results, we conducted experiments with differently sized data sets. We recorded the values of precision and recall for different threshold values for our three data sets with 10% test data (x=10%). We can observe from the results shown in figure 5.9 and our significant tests that the differences between the values are not significant . While the small data set seems to create slightly better results, the large data set shows slightly better precision and recall than the medium data set. Thus, the algorithm is not really sensitive to the data size. For that reason and also because of need of more computation time for larger data set, we used the small data set for the further experiments.

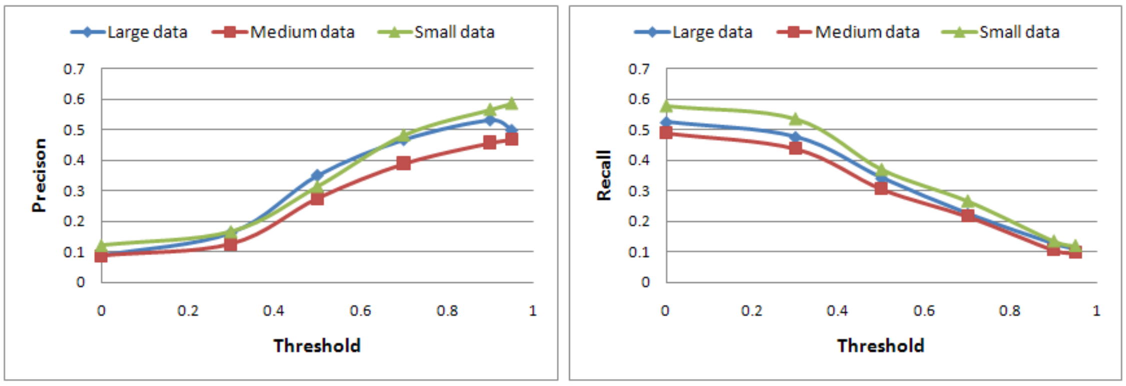


Figure 5.9: Precision and Recall for different threshold values in three data sets with different sizes

Effect of combination factor α

To determine the sensitivity of the value of α for combining different similarity cues, we conducted experiments with different values of α. The result of this experiments is shown in the left chart of figure 5.10. From this chart we select the value of $\alpha = .4$.

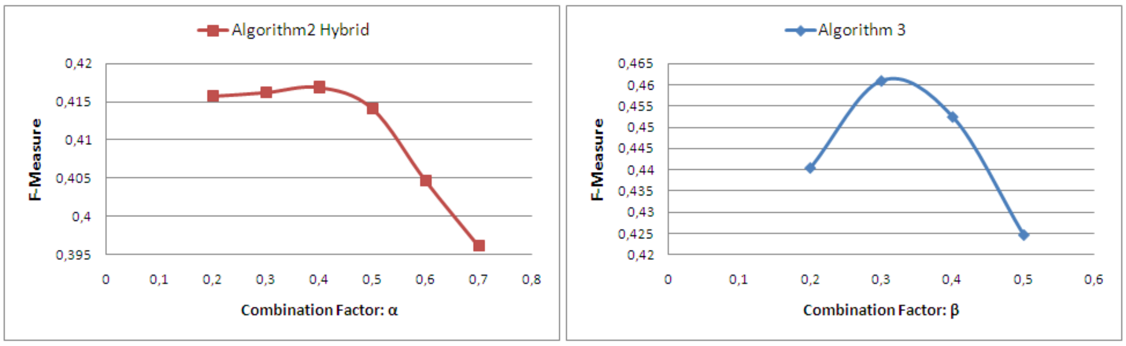


Figure 5.10: Selection of the optimum combination factors

Comparison of similarity measures

Once we obtain the optimal values of the parameters, we compare the performance of the algorithm when using different similarity computation techniques. In addition,we compared our approach with a baseline algorithm suggested in [93]. The results of this comparisons are shown in figure 5.11. The baseline algorithm uses the cosine similarity between concepts and recommends the concepts with highest cosine similarity. Basically, the baseline algorithm is similar to similarity computation step of our algorithm where we find the k-nearest neighbors based on contextual cues. Our results show that the hybrid similarity outperforms other

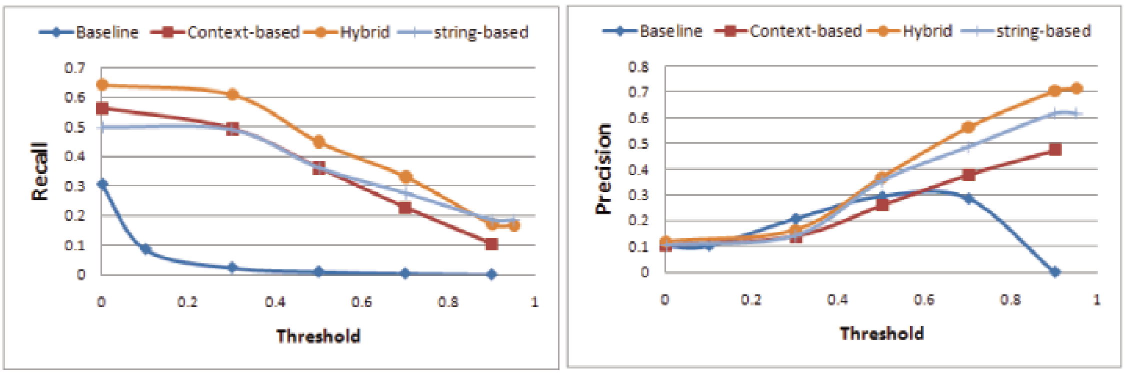


Figure 5.11: Comparison of Precision and Recall of Algorithm 2 when using different similarity measures

approaches for both precision and recall for all threshold and x values. We can observe from figure 5.11 that while contextual cues result in less precision than the string-based techniques, they produce slightly higher recall. That shows that string-based techniques

Wikipedia	Contextual-based	String-based	Hybrid Similarity
Multi-agent systems	Artificial intelligence	Multi-agent systems	Multi-agent systems
Computer network security	Multi-agent systems	Computer network security	Artificial intelligence
	Computer architecture	Computer security organizations	Computer architecture
	Network architecture	Artificial intelligence	Computer network security
	Distributed computing	Computer architecture	Distributed computing

Table 5.2: An example: Output of recommendation algorithm 2 using different similarity cues. Recommendations are predicted super-concepts for the new concept "Botnets"

can suggest more accurate recommendations but they do not have as much coverage as the contextual cues.

An Example

Table 5.6.2 shows an example from the actual recommendation results. The example shows the five top recommendation for potential super concepts of the new concept "Botnets", when using different similarity measures. The most left column shows the actual super-concepts in Wikipedia. We can observe that even though the system considers the recommendations that do not appear in Wikipedia as incorrect and counts them as miss in calculating the precision and recall, in many cases they might actually make sense.

5.6.3 Experimental Results from Algorithm 3

We applied the hybrid recommendation algorithm described in section 5.4.3 by combining the results from the hybrid similarity approach of algorithm 2 and TFIDF string-based technique from algorithm 1. The right chart of figure 5.10 shows the result of the sensitivity analysis for selecting the optimum β. We selected the combination factor, $\beta = .3$ from this chart.

The precision and recall values shown in figure 5.12 show that the performance of algorithm 3 does not have significant difference with algorithm 2 in respect to recall while it can result in considerably higher precision. We can achieve precision of 90% when the threshold value is set accordingly.

Figure 5.13 and 5.14 show comparison of performance of different algorithms. Figure 5.13 shows the F-measure for each algorithm as threshold value changes and figure

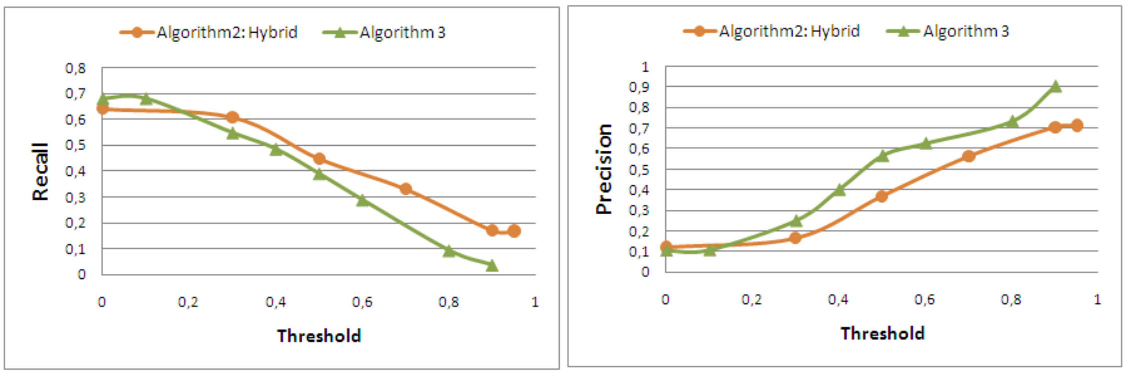


Figure 5.12: Comparison of precision and recall values for algorithm 2 and algorithm 3

5.14 shows recall at N. Basically, we count the correct answers as we recommend N super-concepts. We can observe from both charts that algorithm 3 obviously outperforms all other techniques. In addition, it is clear from the charts that the hybrid similarity outperforms each individual similarity cue. However, it is not trivial to determine if string-based similarity performs better in algorithm 2 than the contextual one. The contextual cues outperform string-based ones when looking at the Recall at N while based on F-measure the string-based techniques outperform contextual ones.

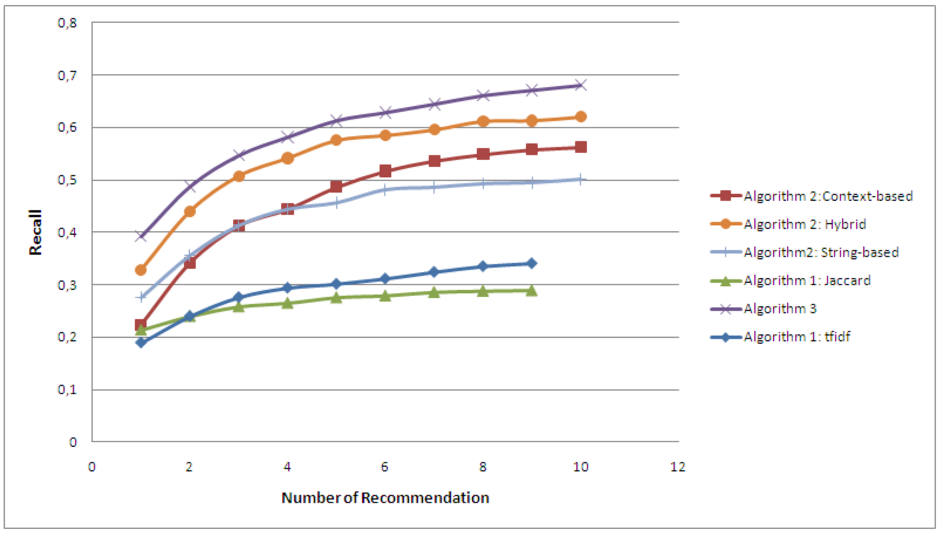


Figure 5.14: Comparison of recall at N for different algorithms

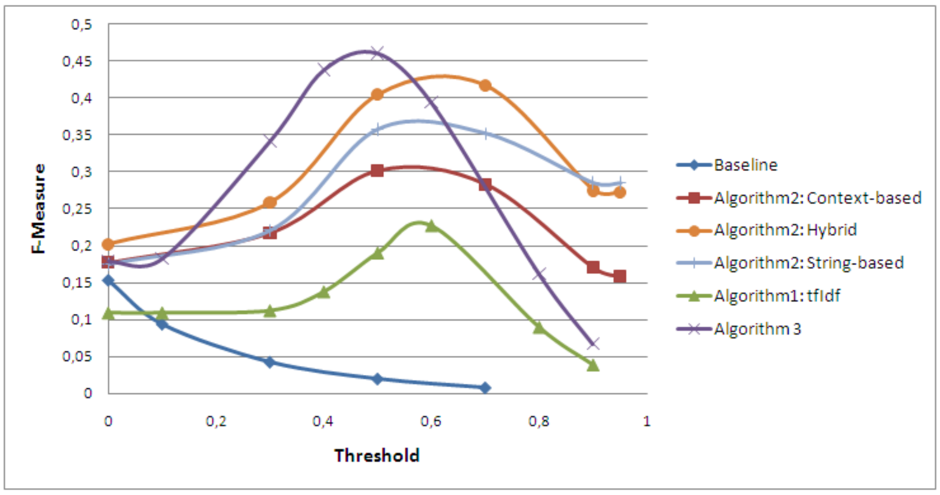


Figure 5.13: Comparison of F-measure for different approaches by changing the threshold value

Impact of the Seed Hierarchy Density

To determine the effect of density of the seed concept hierarchy, we carried out an experiment where we varied the value of x from 20% to 80%. Our results are shown in figure 5.15. As expected, the quality of recommendations of algorithm 2 decreases as we increase x since there is less information for the algorithm to learn from the existing hierarchy. However, the performance of algorithm 1 does not change as we increase x since algorithm 1 does not need training. Algorithm 3 benefits from this feature of algorithm 1 and its performance does not drastically decrease as the seed hierarchy gets smaller. Thus, algorithm 3 can perform well when the system is almost new and there is not much existing hierarchy information to learn from and the performance would increase as more information is entered into the system by the users.

5.6.4 Expert Evaluation

Table 5.3 shows an example of the potential super-concepts suggested by each algorithms for the concept "Computer hardware researchers". We can see that the actual super concepts for this concept in Wikipedia are "Computer scientists by field of research" and "Computer architecture". While only these two concepts are considered as correct answer, we can see that some of the recommendations might be even more relevant than these two concepts. For example, "Computer scientists" and "Computer systems researchers", recommended by

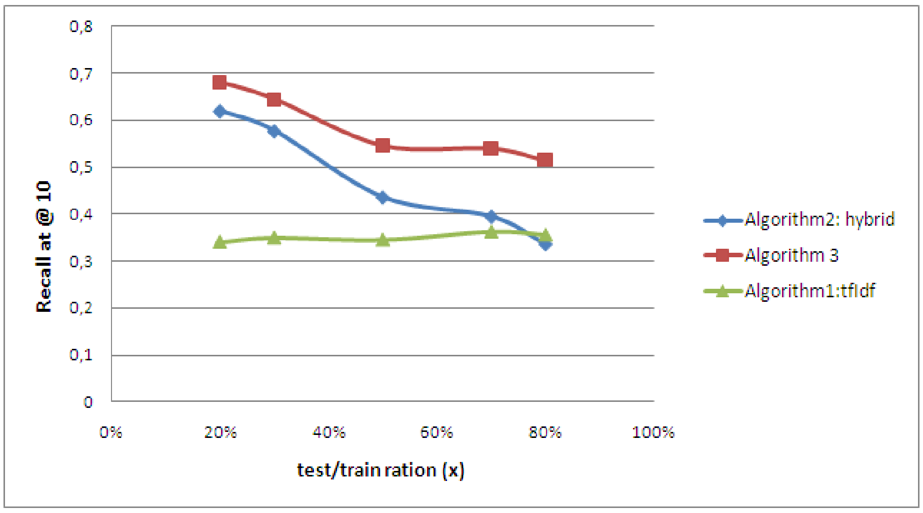


Figure 5.15: Impact of the test/train ratio (the density of the seed hierarchy)

algorithm 2 and algorithm 3 are certainly more relevant to be the super-concept of "Computer hardware researchers" than "Computer architecture".

To gain insight into the quality of the recommendation created by our algorithm, we further performed an expert evaluation by exploring sample recommendation results from algorithm 3. We extracted 100 examples from the data set and asked 10 computer science experts (senior PhD students and postdocs) to evaluate if the recommended super-concepts are appropriate. We provided the experts with two sets of potential super-concepts for each concept. The first set consists of the super categories directly extracted from Wikipedia and the other set consists of the results of our algorithm. Thus, the experts have evaluated our algorithm as well as the Wikipedia category hierarchy.

The results from the expert evaluation is shown in table 5.4. The Wikipedia category hierarchy has precision of 82% which means that in some cases (18% of the cases), the experts thought that Wikipedia super-categories are inappropriate. Since our algorithm learns from exiting relations in Wikipedia, we do not naturally expect to get better results than Wikipedia. However, we can see that based on expert evaluation reported in table 5.4, our algorithm performs even better than the reported values in the charts. For example, in figure 5.12, the precision is equal to 58% when the threshold is set to .5 while in table 5.4, the precision for this threshold is 89%.

We report the average number of super-concepts instead of recall since it is only possible to measure recall if the correct super-concepts are clearly defined in advance. The average number of super-concepts for each concept in Wikipedia is 1.7. We can see that to get

Wikipedia	Algorithm 1: Jaccard	Algorithm 2: Hybrid	Algorithm 3
Computer scientists by field of research	Computer vision	Computer scientists by field of research	Computer scientists by field of research
Computer architecture	Computer architecture	Computer scientists	Computer scientists
	Database researchers	Computer systems researchers	Computer architecture
	Graphics hardware	Computer graphics	Computer systems researchers
	Computer scientists	Computer graphics researchers	Graphics hardware

Table 5.3: An example: Output of different recommendation algorithms. Recommendations are predicted super concepts of the new concept "Computer hardware researchers"

similar coverage from our algorithm (average number of super-concepts=1.65), we should set the threshold to .4 which results in precision of 77%. We also looked at the precision when exactly one super-concept is recommended for each new concept independent of the threshold value. In this case, the average number of super-concept is obviously 1 and it results in precision of 81%.

Threshold	Precision	Average number of super-concepts
.9	100%	.16
.7	97%	.45
.5	89%	.87
.4	77%	1.65
–	81%	1
Wikipedia Hierarchy	82%	1.7

Table 5.4: Expert evaluation results

5.6.5 Discussion

From the experimental evaluation of the algorithms, we make some important observations. Each of the proposed algorithms has certain pros and cons. Algorithm 1 does not require a seed hierarchy and this feature makes it applicable to any kind of system. However, the algorithm is very basic and does not learn or get better as more data enters to the system. Thus, this algorithm might only be appropriate at start phase of a system. On the other hand, algorithm 2 requires a seed hierarchy to start with but it has the advantage that it can learn from the existing hierarchy and also considers the contextual information. Therefore, it results in considerably higher accuracy and coverage. Algorithm 3 can benefit from both algorithms 1 and 2 without having the limitations of each individual algorithm. We can conclude from the overall results that the hybrid algorithm (algorithm 3) can be used as a strong algorithm for recommendation of potential super-concepts independent of the properties of the seed hierarchy.

The qualitative results from expert evaluations show that our proposed algorithm can produce results which are almost as accurate as the existing Wikipedia hierarchy. Thus, the system can also be used directly in Wikipedia for improving the coverage of the category hierarchy.

In this work, we have focused on recommendation of super-concepts. An important feature of our proposed algorithm, is that it is not only limited to super-concepts. Other types of semantic relations such as sub-concepts, synonymy, antonymy, polysemy, etc. can be recommended by defining the SSA value in the right way. For example, to recommend sub-concepts, it is enough to define the $SSA(C1, C2) = 1$ when C1 is sub-concept of C2. Similarly, for any other type of semantic relations, it is enough to define the SSA accordingly.

5.7 Conclusion and Future Work

In this chapter, we presented machine learning algorithms to support continuous ontology development. Our main contribution in this chapter is suggestion of a machine learning algorithm that learns from exiting relations of an ontology. Our extensive experiments with Wikipedia data set and our expert evaluation show excellent results. The important property of our proposed algorithm is that it can easily be adapted for learning other kinds of relations in an ontology.

This work introduces a new direction on semi-automatic collaborative ontology development. There is a lot of opportunity for further investigation in this area. We list several possible extensions to this work.

Contextual similarity In this work, we have used associated pages to a concept as contextual information. Other contextual information such as links between pages, user information, and other contextual cues can be also considered. Understanding and modeling user context is an area of active research and can have many applications in search, personalization and recommendation including recommendation of semantic relations as presented in this work.

Other semantic relations In this work, we have focused on recommendation of super-concepts. Similar techniques can be utilized for recommendation of other types of semantic relations such as synonymy, antonymy, polysemy, etc.

Evaluation Our evaluation methodology removes all the test cases from the ontology and does not consider the temporal aspect of the development of the ontology. The more accurate evaluation would be to add each test case to the training set after it receives a correct recommendation. This evaluation would also show us how the ontology develops over time as users use the system and contribute.

Evaluation of the algorithms in an actual interactive social semantic bookmarking application can help us answer the question whether the generated recommendations are actually perceived as useful by the user.

5.8 Acknowledgment

The work presented in this chapter was initiated by support of SAP Research during my internship at SAP Research, Karlsruhe in summer 2009. With support of FZI (Forschungszentrum Informatik), the simplified version of the algorithms were implemented into SOBOLEO [2] system. Thanks to Hans Friedrich Witschel, Simone Braun, and Valentin Zacharias for their contributions to this work. This work was supported in part by the MATURE Project cofunded by the European Commission.

[2] www.soboleo.com/

Chapter 6

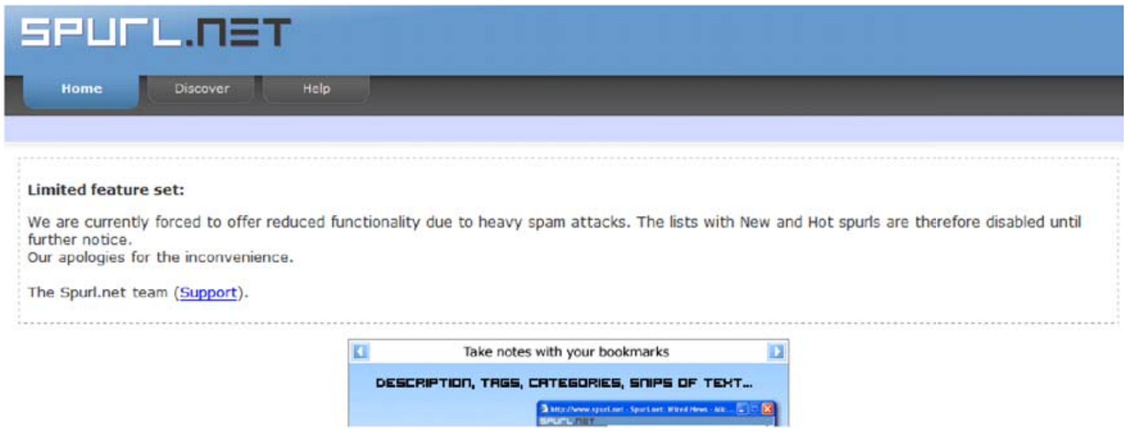
Combating Attacks Against Social Tagging Environments

6.1 Introduction

Security is one of major challenges of open adaptive systems. In this chapter, we address the problem of attacks against social tagging systems by modeling different kind of attacks and evaluating their impact on the systems' behavior. Gaining a fundamental understanding of the nature and impact of such attacks will hopefully lead to more secure and robust social Web applications.

Social tagging environments are gaining popularity in part because they provide an open social environment for users to share resources and opinions and give them the ability to classify information in a natural way. There is typically no limit to the number of tags that may be assigned to a resource and there is no strict hierarchy of tags. Social tagging systems can be considered as an extension of social recommendation behavior: people share resources and tags, which connect them implicitly in a social network. For example, in Delicious or last.fm users can find other users with similar tags or resources and build a network with them. Like other publicly accessible adaptive systems such as collaborative recommender systems, tagging systems present a security problem. Attackers, who cannot be readily distinguished from ordinary users, may inject biased profiles in an attempt to force a system to adapt in a manner advantageous to them.

Recent work has established that adaptive Web applications, such as collaborative recommender systems, can be manipulated via "profile injection attacks". In a profile injection attack (sometimes called "shilling"), an attacker uses fictitious identities to insert biased implicit or explicit ratings into a recommender system [34]. Such profiles may be generated manually by an attacker or an automated agent. These attacks do not require a great deal of


SPURL.NET
Home
Discover
Help

Limited feature set:

We are currently forced to offer reduced functionality due to heavy spam attacks. The lists with New and Hot spurls are therefore disabled until further notice.
Our apologies for the inconvenience.

The Spurl.net team (Support).

Take notes with your bookmarks
DESCRIPTION, TAGS, CATEGORIES, SNIPS OF TEXT...

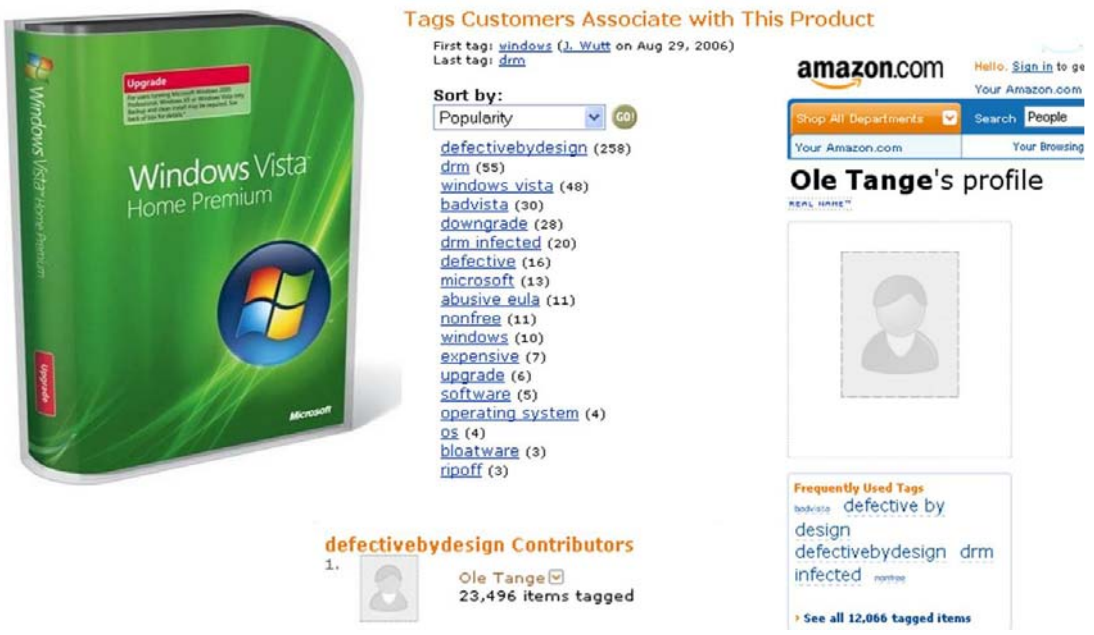
Figure 6.1: Spurl.net response to attacks(screenshot made in 2009)

knowledge about the details of the recommender system or its algorithms [155, 117, 33, 142].

Tagging systems are also dependent on public input, and are therefore susceptible to profile injection attacks. Attackers may use misleading tags to confuse others or to achieve some goal, such as promoting or demoting a product or brand.

There have been numerous real-world examples of suspicious behavior in social tagging systems. For instance, Delicious popular, a collection of most popular URLs in the delicious bookmarking service, has been reported spammed by a malicious user [129]. In Flickr, users have reported spam tags on their personal photos[208]. Another social bookmarking Website, “Spurl.net”, had to reduce functionality due to spam. Figure 6.1 shows an screenshot of the Website response to spam attacks. Hundreds of such examples can be found by simply browsing popular social Websites. Figure 6.2 shows an attack against Windows Vista in the popular e-commerce Website, Amazon.com. The most popular tag for this product is “defectivebydesign” which has been associated to this product 258 times. We looked the list of users who have used the tag “defectivebydesign” in their profile and we observed that the user “Ole Tange” is one of the users who have mostly used this tag and this user has 23,496 items tagged. Looking at the list of tags is his profile, we noticed that his most frequent used tags are “defectivebydesign”, “defective by design”, “infected”, and “drm”. This is an example of a goal-oriented attack to demote a target product.

The above examples underline the importance of security in social tagging sites. To analyze the vulnerability of a tagging system, we must first understand the nature of the attacks against it. We are primarily interested in attacks where an attacker’s aim is to maliciously influence the system. For instance, consider the example shown in Figure 6.3 of a tagging system that allows users to annotate URLs. A subset of tag assignments are displayed for users (User1 - User6). Suppose a user is searching for the resource that is most related to



Tags Customers Associate with This Product

First tag: windows (J. Wutt on Aug 29, 2006)
Last tag: drm

Sort by: Popularity

defectivebydesign (258)
drm (55)
windows vista (48)
badvista (30)
downgrade (28)
drm infected (20)
defective (16)
microsoft (13)
abusive eula (11)
nonfree (11)
windows (10)
expensive (7)
upgrade (6)
software (5)
operating system (4)
os (4)
bloatware (3)
ripoff (3)

defectivebydesign Contributors

1. Ole Tange
23,496 items tagged

Ole Tange's profile

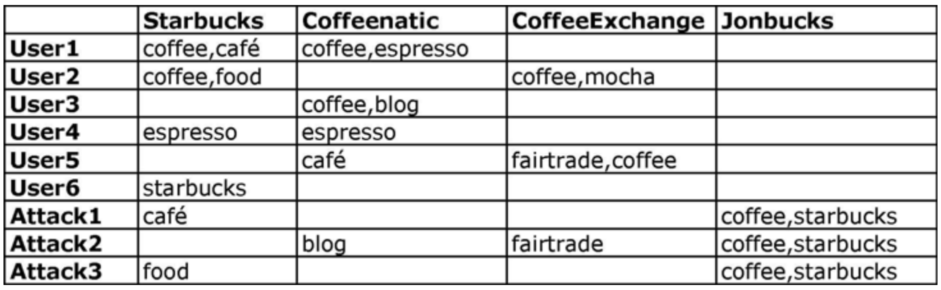
Frequently Used Tags
badvista defective by design defectivebydesign drm infected nonfree

› See all 12,066 tagged items

Figure 6.2: An attack example in Amazon.com (screenshot made in 2010)

the tag "coffee". Prior to attack, the system will display the resource "Starbucks" based on the number of occurrences of the tag. Now suppose another coffee shop, "Jonbucks", wishes to promote the resource "Jonbucks" to a segment of users interested in coffee. Attack profiles (Attack1 - Attack3) are created, assigning the tag "coffee" and "starbucks" to "Jonbucks" webpage. After the attack, the system displays Jonbucks Webpage as the most related resource to users who search for tags "cofee" or "starbucks".

Our goal in this chapter is to systematically study the problem of attacks against social tagging systems and utilize data mining approaches to address this problem. In this chapter, we first present a set of attack dimensions that establish a context for our analysis. We next discuss attack types, based on navigation channels and attack targets within a tagging system. These attack types represent in abstract the strategies that could be employed by attackers in order to manipulate the output of the system. We present experimental results using real world data sets to show the system's vulnerabilities using several proposed evaluation metrics. Finally, we conclude the chapter with directions for future work.



	Starbucks	Coffeenatic	CoffeeExchange	Jonbucks
User1	coffee,café	coffee,espresso		
User2	coffee,food		coffee,mocha	
User3		coffee,blog		
User4	espresso	espresso		
User5		café	fairtrade,coffee	
User6	starbucks			
Attack1	café			coffee,starbucks
Attack2		blog	fairtrade	coffee,starbucks
Attack3	food			coffee,starbucks

Figure 6.3: An hypothetical example of promoting a resource.

6.2 Related Work

Previous studies on profile injection attacks against collaborative filtering recommenders [143, 155, 117, 33, 142] have examined the effects of attack types that require varying degrees of knowledge about the system, and proposed several responses to attack. The outcome of this research produced guidelines for developing more robust alternatives to standard collaborative filtering algorithms and it demonstrated that a number of model-based and hybrid algorithms offer substantial improvement over standard algorithms [185, 144]. Another proposed response is to detect and defeat attackers before they cause harm [164]. Supervised classification approach are used to identify and respond to profile injection attacks [38].

Researchers have begun to study attacks on social tagging systems. Xu et al. [224] have introduced basic criteria for a good tag and proposed a collaborative algorithm for suggesting tags that meet these criteria. From their perspective, a set of tags that is used by a large number of people for a particular item is less likely to be spam and thus those tags are considered as high quality tags for that item. The authors have accounted for spam by assigning a reputation score to each user, based on the quality of the tags contributed by that user. These reputation scores have been used for identifying good candidate tags for a particular document, i.e., for automatic tag selection. In the proposed tagging system, a new user has to get a certain reputation score to be able to have impact on the system and to get such reputation score, the user needs to associate the typical tags that others have associated to an item. While such system might be able to produce accurate tag recommendation for many users, it has several problems. First, tags that are considered to be high quality are normally very general and ambiguous. Second, attackers can get a high reputation score by using most popular tags for certain items and then they can influence the system behavior to their favor.

Koutrika et al. [111] have proposed an ideal tagging system where malicious tags and malicious user behaviors are well defined. They propose a trusted moderator who periodically checks if user postings are "reasonable". The moderator also identifies good and bad tags for any resource in the collection. The authors have also defined several query schemes and moderator strategies to counter tag spam.

Heymann et al. [92] surveyed three categories of potential countermeasures: those based on detection, demotion, and prevention. Although many of these countermeasures have previously been proposed for email and Web spam, the authors found that their applicability to social Web sites differs. The authors identify four characteristics of social tagging systems that change the dynamic of the adversarial relationship between service providers and spammers. These characteristics are: (1) One controlling entity who owns and manages the system's content and maintenance (2) Well-defined interactions determined by the controlling entity; For example, a social bookmarking system lets users contribute bookmarks annotated with tags, and possibly share bookmarks with others, but few other actions might be allowed (3) Content and actions in the system are tied to user identifiers (4) Multiple interfaces and views are available to browse the site content. We discuss these multiple views in the next section when we introduce the navigation channels in social tagging systems.

Krause et al. [113] use classification techniques to detect spam in Bibsonomy. They introduce 25 features divided into four categories for classification purpose. The categories include profile-based, location-based, activity-based, and semantic features. Experimental results on Bibsonomy data set show that classical machine learning techniques although not perfectly but can help to solve the problem. The difference of this work with classical web spam classification is the features applied since more information such as email and tags are available. In addition, authors found out that spammers reveal their identity by using similar vocabulary and resources. Thus, it has been suggested that co-occurrence features tackle the problem very well.

Prior work in this area, however, does not generally take into account the goals of an attacker, the audience of the attack, and the context in which the attack is mounted. Spammers might have very different goals by spamming Bibsonomy, which is focused on scientific articles than more broad systems such as Delicious which captures different kinds of domains and a broader audience. Our work, in contrast, considers those aspects and is more concentrated on the practical aspects of how existing systems handle large-scale profile injection attacks. We are interested in looking at this problem from a more practical perspective and try to determine what might motivate an attacker to spam the system.

The goal of our research is to answer the following research questions. What attack types are more successful? Which attack targets are more vulnerable? How many malicious users can a tagging system tolerate before results significantly degrade? How much effort and knowledge is needed by an attacker to attack the system? How can we detect attacks and protect the system? To answer such questions we first characterize the various realistic attack scenarios. We believe that studying properties of typical attack strategies will lead to improved detection algorithms and to robust retrieval algorithms. We empirically examine attack strategies using real-world tagging data towards exposing general vulnerabilities.

6.3 Navigation Channels in Social Tagging Systems

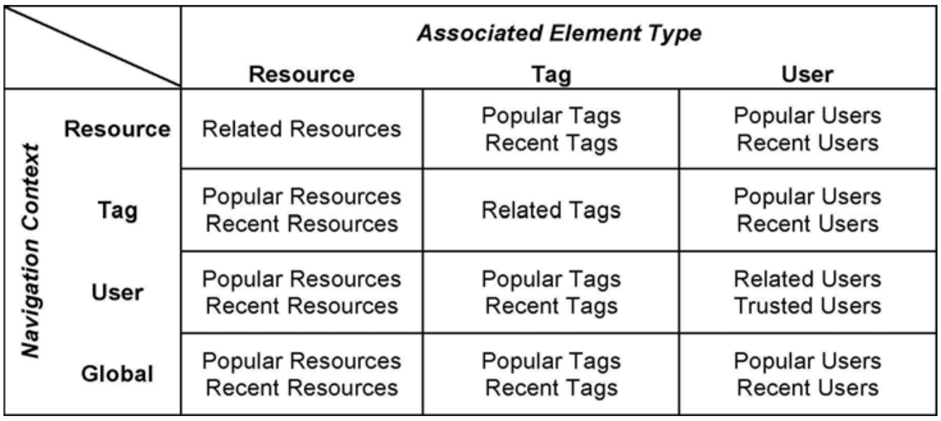
The success of collaborative tagging is partially due to facilitating the retrieval and discovery of resources within a single user-centric environment. Many tagging systems publicly display each user's tags and resources, making retrieval of previous annotations both simple and intuitive. However, the discovery process is much more complex. Users browse the social tagging graph via the many associations between resources, tags, and users. This ability to navigate through the folksonomy is one reason for the popularity of collaborative tagging but it also provides attackers with many possibilities to attack the system.

Understanding the avenues for attacking a social tagging system requires analysis of its navigation process. However, there has been little formalization of tagging system outputs, and much research treats tagging systems solely as retrieval engines, ignoring the flexible browsing environment such sites offer. There is a need therefore for a general model of navigation options and system outputs that can help us model the impact that an attacker may have.

It is beneficial to distinguish the roles of interaction between the annotation and navigation processes. In particular, annotation is concerned with a contributor to the tagging system, whereas navigation is concerned with the viewer. There is no requirement that the viewer of a tagging system is also a contributor. Although it is often the case that contributors annotate resources for their own consumption, most tagging systems also allow unregistered visitors to browse. For example, users of Delicious typically annotate their bookmarks for personal consumption, but anyone can browse the site [82].

We described our terminology for modeling tagging systems in chapter 2. Briefly, the model can be described as a four-tuple $D = \langle U, R, T, A \rangle$, such that there exists a set of users, U; a set of resources, R; a set of tags, T; and a set of annotations, A. Annotations are represented as a set of triples containing a user, tag and resource such that $A \subseteq \{\langle u, r, t \rangle : u \in U, r \in R, t \in T\}$. Each combination of element types R, U, and T represents a specific navigation channel for presenting information in a tagging site. The context of a channel is a reference point for retrieving associated elements. In particular, it is the specific $r \in R$, $t \in T$, or $u \in U$ that serves as a query. Many tagging systems will also include a global context, with no specific query, that facilitates exploration of the site. Given a context, the system will return a set of associated elements of a specified type that are relevant to the context.

As an illustration, consider the Tag-Resource channel. Conceptually, we consider the Tag-Resource channel from an information retrieval perspective. Viewing the reduced bipartite graph TR as a corpus, we map R and T to documents and terms, respectively. The



		Associated Element Type		
		Resource	Tag	User
Navigation Context	Resource	Related Resources	Popular Tags Recent Tags	Popular Users Recent Users
	Tag	Popular Resources Recent Resources	Related Tags	Popular Users Recent Users
	User	Popular Resources Recent Resources	Popular Tags Recent Tags	Related Users Trusted Users
	Global	Popular Resources Recent Resources	Popular Tags Recent Tags	Popular Users Recent Users

Figure 6.4: Navigation Channels of a Tagging System

channel is represented as a single-term query, such that the tag t_q is the user's current tag context. The query returns the most relevant resources $R_t \subset R$ that have been annotated with t_q.

Other navigation channels can be specified in a similar manner, as shown in Figure 6.4. "Related tags" can be calculated by the system based on the number of co-occurrence of tags or other criteria. For example, if the tags "ontology" and "semantic" have been associated to many resources together, they can be considered as related. Similarly, "related resources" can present other resources that have similar tag profile. A tagging system may choose to include only a subset of the possible channels: for example, delicious offers the related tags but does not have a related resource function. The information that is displayed, however, will fall into one of the channels described here. This model allows a common analysis of different systems.

6.4 Attack Dimensions

In this section, we present seven dimensions of an attack against a social tagging systems. Specifically, we discuss motivation of the attacker, intent of the attacker, generality of the intended audience, degree of profile obfuscation, size of attack, navigation context, and target element. These dimensions are based on our work presented in J.J. Sandvig et al. [186] and are identified by contributions and help of my colleague J.J. Sandvig.

Motivation of Attacker

At a basic level, an attacker may be motivated to either disrupt the tagging system as a whole, or to promote a particular viewpoint within the system. In the first case, an “eBully” may attempt to introduce random noise into the system, simply to promote anarchy or to degrade the reputation of the system. Although certainly a concern, it is difficult to quantify an attack motivated by disruption because of the subjective decision about when an outlier is considered true noise and when it is considered an attack.

Our primary focus is on the attacker interested in promoting a particular viewpoint. Presumably, the attacker wants to bias the system in order to produce some economic or political advantage. Furthermore, the viewpoint may include a short-term or long-term purpose. For example, a political activist or special interest group may have a short-term goal of influencing a particular vote, or a long-term goal of promoting some larger issue. Likewise, a firm may attempt to manipulate a market in the short-term for economic gain or have a long-term goal of promoting a particular product or brand.

Intent of Attacker

If motivation describes the “why” of an attack, then an attacker’s intent describes the “what”. It is the desired outcome of a particular attack campaign. The tagging system may be the direct target of attack, or it may be used indirectly to influence the actual target of attack.

In a direct attack, the intent may be to promote a particular product within the tagging system itself, or to demote a competitor’s product. We call these “push” and “nuke” attacks, respectively. In an indirect attack, the intent is to use the tagging system platform in order to bias some other system. For example, an attacker may use a social bookmarking system to create a large number of back-links to some target URL, in an attempt to raise its Google PageRank value.

Intended Audience

It may not always benefit an attacker to throw the widest possible net. Instead, an attack is likely to be aimed at those users of the system that are most receptive to the overall intent. For example, in Figure 6.3, “Jonbucks” coffee shop is attempting to promote its Web site on a social bookmarking system. The company’s goal might be to improve its ranking with respect to those users that are interested in coffee, a targeted-marketing strategy.

The generality of a targeted user segment may be different, depending on the context of the attack. The intended audience may range from universal to focused. An attack on a completely generic user segment is analogous to finding the lowest common denominator within the entire user community – attempting to promote a product to the most common and popular interests.

As an illustrative example of the difference between universal and focused attacks, look again at an attack to promote Jonbucks coffee shop. To target all users, Jonbucks would annotate its site with the most popular tags in the entire tagging system, regardless of their relevance: "design" and "blog" are the most popular tags on Delicious at the time of writing this chapter. For targeting a coffee-focused user segment, Jonbucks would use tags such as "coffee" and "mocha", which are likely to be of employed by those users. For our purposes, we will consider an attack that uses the most popular tags to be a general attack. An attack using any other set of tags is assumed to be a "focused attack" directed towards the users who tend to employ those tags.

Degree of Attack Profile Obfuscation

Depending on the intent, an attacker may obfuscate the injected user profiles to help mask the attack. In an extreme example, great care may be taken to ensure that an attack profile looks exactly like a real user profile. The attacker tries to mimic an expert in the domain of the targeted user segment, building trust until the attack is carried out.

On the other end of the spectrum, the attacker doesn't care if the profile looks legitimate at all, and focuses only on maximum effect in biasing the system's retrieval algorithms. The degree of profile obfuscation is a tradeoff, as greater obfuscation is more difficult for the system to detect, but is more labor intensive to build and takes longer for the attacker to see returns.

Size of Attack

The size of attack measures the number of coordinated attack profiles that are added to the tagging system. The minimum number of profiles required for an attacker to obtain the desired effect is largely influenced by the overall goal of the attack. If the goal is to mimic a domain expert, the attack may be successful by using only one or two carefully constructed user profiles.

However, if the goal is to bias the system's retrieval algorithms, a large number of attack profiles may be necessary in order to bias the aggregate ranking of the attack target, relative to related elements. In this case, the popularity of related elements has a large effect on the size of a successful attack. As an illustration, look again at the Jonbucks attack on the tag "coffee". If there are very few bookmarks that are tagged with coffee, then relatively few attack profiles need to be created that annotate Jonbucks with coffee. However, if "Starbucks" has already been tagged with coffee over 100,000 times, then Jonbucks has a much larger hurdle to clear, requiring a very large number of attack profiles to surpass the popularity of Starbucks.

Navigation Context

Navigation context refers to a specific resource, tag, or user in the tagging system that provides a mechanism for navigating its associated elements. It is the current location of a viewer who is browsing or querying the system. An attacker may focus on a particular navigation context as the reference point of attack. In the Jonbucks example, the tag "coffee" is the navigation context, and the attacker wants to improve the rank of the Jonbucks Web site within that context.

An attack may include multiple navigation contexts (e.g., Jonbucks might utilize both "coffee" and "mocha" tags). However, for the purposes of this paper we will focus on attack using a single context. This does not mean, however, that an attack aimed at a single context will only impact one aspect of the tagging system. In the Jonbucks example, attacking the "coffee" tag context may have the unintended result of making Jonbucks and Starbucks very similar resources. If the tagging system includes a navigation channel for displaying similar resources, someone viewing the Starbucks resource may then see Jonbucks ranked highly.

Target Element

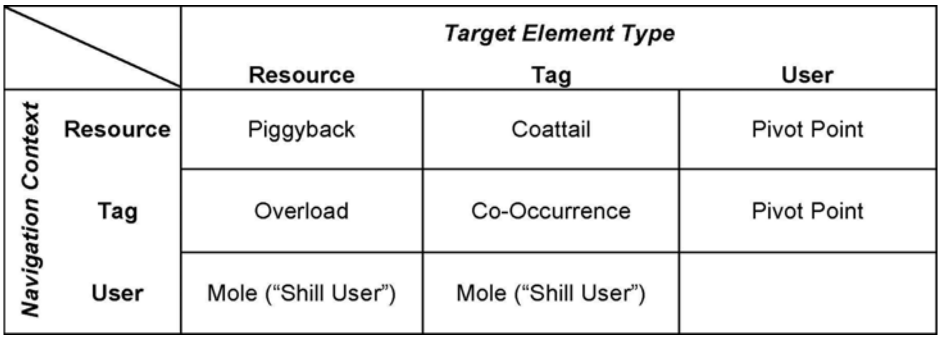
Target element refers to the specific resource, tag, or user in the tagging system that is the actual target of attack. It is the element that the attacker wishes to promote or demote. In many cases, this is likely to be a resource. In the Jonbucks example, the attacker wants to improve the visibility of the Jonbucks Web site.

However, the target element could also be a tag or user. An attacker may want to push the tag "Jonbucks", simply to raise brand awareness. The tag could be associated to the tag "coffee" such that Jonbucks is advertised as related to coffee, or the tag could be annotated to the resource "Starbucks" as an alternative brand. Similarly, an attacker may want to push a personal user profile as a form of self-promotion.

6.4.1 Attack Types

An attack type is a strategy for building attack profiles. Studying attack types allows us to classify common patterns of attack and identify their aims and tactics. Our categorization of attack types is based on the navigation channels shown in Figure 6.4. Figure 6.5 summarizes the types.

An attack type is a generic strategy for building attack profiles. It is a partial model based on abstract navigation context and target element types. A particular implementation



		Target Element Type		
		Resource	Tag	User
Navigation Context	Resource	Piggyback	Coattail	Pivot Point
	Tag	Overload	Co-Occurrence	Pivot Point
	User	Mole (“Shill User”)	Mole (“Shill User”)	

Figure 6.5: Summary of Attack Types

of an attack type includes specific details, and should be analyzed according to the attack dimensions introduced in Section 6.4. However, studying generic attack types allows us to classify common patterns of attacks at a strategic level. We now propose a number of attack types that correspond to the different navigation channels within a social tagging system. A summary of attack types is shown in Figure 6.5.

Overload

[Context: tag. Target: resource] The goal of an overload attack, as the name implies, is to overload a tag context with a target resource so that the system correlates the tag and resource highly. The assumption is that the attacker wants to associate the target resource with some high-visibility tag, thereby increasing traffic to the target resource. If the intended audience of the attack is general, a popular tag is chosen. If the intended audience is specific, a focused tag is chosen that is particular to the targeted user segment.

Piggyback

[Context: resource. Target: resource] The goal of a piggyback attack is for a target resource to ride on the success of another resource. It exploits the idea of sharing tags among resources, attempting to associate the target resource with some resource context, such that they appear similar. The resource context may be popular or focused, depending if the intended audience is generic or specific.

There are two possible implementations of piggyback. The *tag duplication* technique is to pick a number of tags highly correlated to the resource context and annotate the target resource with the same tags, preferably with the same distribution. The *tag overlap* tactic is to pick any number of random tags and annotate both the resource context and the target resource with those tags within the same attack profile.

Coattail

[Context: resource. Target: tag] The goal of coattail is for a target tag to be correlated with a particular resource context. The resource context may be popular or focused, depending if the intended audience is generic or specific, respectively. An attack is created by annotating the resource context with the target tag in every attack profile.

For example, an attack can associate the tag "Jonbucks" to the resource "Starbucks". By creating multiple attack profiles, the Jonbucks tag may be pushed to the top of the list of popular tags for Starbucks URL, making it highly visible to users looking for tags associated with Starbucks.

Co-occurrence

[Context: tag. Target: tag] The goal of co-occurrence is for a target tag to be correlated with another popular or focused tag. An attack consists of annotating any resource with both tags, such that they always occur together. The assumption is that the attacker wants the target tag to show up as a "related tag" to the tag context. Tagging systems that measure the similarity between tags may increase the rank of the target with respect to the tag context. A user that views the tag context will have a high chance of seeing the target in the list of related tags.

There are two possible implementations of co-occurrence. The *resource duplication* technique is to pick a number of resources highly correlated to the tag context and annotate each resource with the target tag, preferably with the same distribution as the tag context. The *resource overlap* technique is to pick any number of random resources and annotate them with both the tag context and the target tag within each attack profile.

Mole

[Context: user. Target: resource or tag] The goal of mole (or "shill user") is to create profiles intended to build trust within a targeted audience. The audience may be general, or more likely, a focused user segment. Over time, the attack profiles annotate resources relevant to the targeted audience in such a way as to mimic a domain expert. At some point after the attack profile has established trust, the intended target resource or tag is injected into the profile, hoping that other users in the segment will simply assume it is also relevant to them.

Pivot Point

[Context: resource or tag. Target: user] The goal of pivot point is to create a strong association between an attack profile and its intended audience by correlating it with resources and/or tags that are relevant to the targeted user segment. The user segment may be generic or focused, which determines the choice of resources and tags in the attack profile.

A mole attack may utilize a pivot point in order to establish the attack profile as an expert in the particular user segment. However, pivot point may be used in any general scenario where attack profiles are meant to be highly visible, with the hope that the profiles will receive more traffic. The defining characteristic of a pivot point attack is an indirect link to the actual target element – the attacker wants to raise the visibility of the attack profile itself, which in turn contains the target resource or tag.

6.5 Retrieval Algorithms

In this section, we briefly review the typical retrieval algorithms used in social tagging systems. Within each navigation channel, a retrieval algorithm defines the particular elements considered relevant to the context. Relevance may be displayed in different ways between contexts, such as "popular tags", "recent tags", "recent resources", "active users", "related tags", etc. Generally, results are based on popularity or recency. Some applications may also allow the viewer to choose the appropriate ranking algorithm. For example, delicious allows a user to view the most popular or most recent resources that are annotated with the specified tag.

While other retrieval models may be used, our work focuses on the vector space model [183] adapted from the information retrieval discipline to work with social tagging systems. The following equations assume retrieval is based on the Tag-Resource channel using the reduced TR bipartite graph; however, they may be easily modified to support retrieval in any navigation channel by using an appropriately defined bipartite graph.

A resource vector is represented as $\vec{r} = [w_{t1}, w_{t2}, \cdots, w_{tn}]$ such that w_t is the weight of a particular tag $t \in T$. Vector weights may be derived by many methods, including frequency or recency. In this work, we will rely on frequency. The *tag frequency*, *tf*, for a tag, $t \in T$, and a resource, $r \in R$ is the number of times the resource has been annotated with the tag. We define *tf* as:

$$\textit{tf(t,r)} = |\{a = \langle u,r,t \rangle \in A : u \in U\}| \tag{6.1}$$

Likewise, the well known *term frequency* $\times$ *inverse document frequency* [182] can be modified for social tagging systems. The *tf.idf* multiplies the aforementioned frequency by the importance of the tag t. The importance is measured by the log of the total number of resources, N, divided by the number of resources to which the tag was applied, n_t. We define *tf.idf* as:

$$\textit{tf.idf(t,r)} = \textit{tf(t,r)} \times \log(N/n_t) \tag{6.2}$$

With either term weighting, a similarity measure between a query, q, represented as a vector of tags, and a resource, r, can be calculated. Cosine similarity is a popular similarity measure defined as:

$$cos(q,r) = \frac{\sum_{t \in T} tf(t,q) \times tf(t,r)}{\sqrt{\sum_{t \in T} tf(t,q)^2} \times \sqrt{\sum_{t \in T} tf(t,r)^2}} \tag{6.3}$$

However, in this work we assume navigation is often initiated by selecting a single tag as the navigation context. Therefore, a query is a vector with only one tag, reducing *cos(q,r)* to simply *tf(t,r)*. After similarity is calculated between the query and each resource, an ordered list can be returned to the viewer.

Three navigation channels are symmetric in nature – the Resource-Resource, Tag-Tag and User-User channels. Each has navigation context and target belonging to the same element type, and the outputs of these symmetric channels are based on similarity. For example, Delicious shows a list of tags that are related to a particular tag. In the same way, it is possible to list similar resources and users. In our experiments we use cosine similarity to find related resources and tags. This allows us to evaluate the local effects of Piggyback and Co-occurrence attacks, respectively.

6.6 Evaluating Impact of Attacks

In this section we present our methodology to evaluate the impact of different attack types. We approach the problem with a combination of theoretical modeling, empirical examination of data sets, and simulation of user interactions with social tagging systems.

We are interested in evaluating both the local impact of an attack on a single navigation channel as well as capturing the overall global impact of an attack. By studying localized tagging system output, as defined by a navigation channel, we can see how an end user of the system is actually affected by attack. By studying the global changes to the underlying annotation structure, we are able to gain insight into how attacks propagate through the tagging system.

6.6.1 Measuring the Local Impact of Attack

From the attacker's perspective, an attack is successful if it generates the desired visibility for the targeted element within the intended navigation channel for the targeted audience. Therefore, it is necessary to have localized metrics showing how end users that are browsing the channel are affected by the attack. We can track these channel-specific effects by looking at the rank of the target item before and after the attack with regard to specific retrieval algorithm. We use two metrics for measuring the local impact of an attack: rank improvement and hit ratio.

Rank Improvement

Although the average rank treats differences at the top and bottom of the list identically, from the attacker's point of view a difference between a rank of 10 and a rank of 20 is far more significant than the difference between a rank of 110 and 120. For this reason, we measure the difference in reciprocal rank before and after attack. Let r be the rank of the target item before the attack and r' be the rank afterwards. Rank improvement is given by

$$Imp = \frac{1}{r'} - \frac{1}{r} \tag{6.4}$$

and is relative to the navigation channel, making the local metric specific to the implemented retrieval algorithm.

Hit Ratio

Hit ratio is a measure that allows us to evaluate the changes in the system from a user perspective. We consider the top n results of the retrieval algorithm. Hit ratio is equal to 1 if a particular attack target appears in the top n results, and it is equal to 0 otherwise.

$$Hit\ Ratio = \begin{cases} 1 & \text{If } r_t \in R_n \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

Where R_n is the list of top n retrieved results and r_t is the attack target.

6.6.2 Measuring the Global Impact of Attack

Although local metrics based on navigation channel output are useful for evaluating how end users are affected by attack, they are limited because they only measure attack effectiveness relative to specific navigation channels and retrieval algorithms. A global metric that observes changes in the underlying network of annotations allows us to study attack effectiveness in a more holistic manner. Annotations belonging to an attack create links between the target element and other contextual elements. This propagates to subsequent nodes via other annotations, and so on. Because there are many number of paths to the attacked element that are beyond the attacker's control, there may be unexpected effects of an attack across navigation channels.

In [170] we introduced “Hit Probability” to approximate the probability that a user choosing tags randomly would encounter a given target resource. Although useful, this measure cannot be adapted for other navigation channels or attack types. Thus, we proposed using the Adapted PageRank algorithm for evaluating the global impact of attack in [171]. Adapted PageRank algorithm is described in detail in section 4.3.1. We provide a brief description here as well.

In PageRank, the authority of a page p is defined based on the number of incoming links and on the authority of every page $q \in Q_p$ that connects to p with a forward link [19] and can be calculated as follows:

$$\vec{x} = dW\vec{x} + (1-d)\vec{v} \tag{6.6}$$

Where W is the transition matrix. The elements of W are defined as $w_{i,j} = 1/\left|H_j\right|$ if there exists a link from j to i and $w_{i,j} = 0$ otherwise. $\left|H_j\right|$ is the *outdegree* of j, so every column adds up to either 1 or 0, making the matrix W column stochastic.

The folksonomy adapted PageRank uses the same equation on the weighted tripartite graph, G, with set of nodes $V = U \cup R \cup T$ and three types of edges $E' = E_{u,r} \cup E_{u,t} \cup E_{r,t}$. The weight of each edge $E_{u,r}$ is defined as $|\{\langle u,r,t\rangle \in A : t \in T\}|$, each edge $E_{u,t} = |\{\langle u,r,t\rangle \in A : r \in R\}|$, and each edge $E_{r,t} = |\{\langle u,r,t\rangle \in A : u \in U\}|$. The folksonomy transition matrix is built by normalizing each column of the matrix to 1.

The damping factor d determines the influence of $\vec{v}$, which is typically defined as $v = [1, \cdots, 1]^T$ to make the system randomly jump to another link from time to time but it may be personalized with user preferences. In all our experiments except the focused attacks we consider d as 1 since we are not interested in the random jumps of the system. In the focused attacks, the preference vector is set to target specific group of users.

We believe PageRank can accurately measure global impact because it introduces a notion of page authority that is independent of content and based solely on the graph structure. In the context of tagging systems, a resource that is annotated by authoritative users with authoritative tags can be considered to be authoritative itself. An attack exploits this model because it links the target element to contextual elements and creates a mutual reinforcement of authority that is measurable by PageRank.

To study the overall effectiveness of an attack we measure the Adapted PageRank of an attacked item, then calculate the improvement in a similar manner as defined in section 6.6.1. In this case, the rank of the target item refers to the relative rank of the element based on its PageRank score across all elements.

6.7 Experimental Results

In this section we present results showing the impact of several different attack types described in section 6.4.1. For each attack type, we generate a number of attack profiles and insert them into the system database, testing the effects of different attack sizes and number of selected tag contexts.

6.7.1 Experimental Setup

We provide an extensive evaluation using data from two large real world social tagging systems: Bibsonomy, and Delicious. We use the same data set used in chapter 4. The characteristics of the datasets can be found in table 6.1.

Folksonomy	**Delicious**	**Bibsonomy**
Number of Users	7,665	402
Number of Resources	15,612	2,014
Number of Tags	5,746	1,755
Number of Posts	720,788	15,760
Number of Annotations	2,762,235	53,554
Average Resource Frequency	46.1	7.8
Average Tag Frequency	126.6	10.3
Resource Frequency Std. Deviation	47.6	4.6
Tag Frequency Std. Deviation	461.4	15.8
Resource Frequency Kurtosis	50.2	23.1
Tag Frequency Kurtosis	154.5	25.5

Table 6.1: Properties of the Datasets

To reduce noise we remove the long-tail part of the data by applying p-core to insure that each user, resource and tag appear in at least p posts as in [16, 105]. A post is defined as a user, resource and all tags applied by that user to that resource. Similar to our experiments in section 4.5.1, the chosen p value for Delicious and Bibsonomy is 20 and 5 respectively.

Tag frequency and resource frequency refer to the number of users who have used that tag or resource. Average and standard deviation of tag frequency and resource frequency are shown in table 6.1. Figure 6.6 shows the distribution of tag frequency in Delicious and Bibsonomy data. We can observe that the distribution of the number of users who tag a URL follows a power law, in which a relatively small number of URLs are tagged with high frequency while all the rest occur with low frequency. We use "Kurtosis" to measure the "peakedness" of the distribution. Higher kurtosis means more of the variance is the result of infrequent extreme deviations, as opposed to frequent modestly sized deviations. We can observe that the kurtosis is considerably higher for tag frequency in Delicious data set which implies that the long-tail distribution is more extreme in Delicious tags.

One of our goals is to determine whether tagging distribution of the target object influences attack effectiveness. After p-core pre-processing, we divide the remaining part of data into two partitions and run experiments on each partition independently to explore if the different parts of the distribution show different behaviors.

We use the coefficient of variation ($CV = stdev/mean$) to determine the partition boundaries as described in [178]. CV is a statistical measure of the dispersion of data points in a data series around the mean. CV is a useful statistical measure for comparing the degree of variation from one data series to another, even if the means are drastically different from each other.

```
Input: Frequency distribution of objects in ascending order
Output: $P_1$, $P_2$ The partitions
$FREQ = \{F_1, ..., F_n\}$, Frequency scores
$bin_1 = \emptyset$ , $bin_2 = FREQ$
$AVG_i = avg(bin_i)$;
$STD_i = std(bin_i)$;
Calculate $CV_1 = \frac{STD_1}{AVG_1}$;
Calculate $CV_2 = \frac{STD_2}{AVG_2}$;
foreach $F_j \in FREQ$ do
    while $CV_2 > CV_1$ do
        $bin_1 = bin_1 \cup F_j$
        $bin_2 = bin_2 - F_j$
        Calculate $CV_1 = \frac{STD_1}{AVG_1}$
        Calculate $CV_2 = \frac{STD_2}{AVG_2}$;
    end
end
```

Algorithm 1: Partitioning Algorithm

Data	Delicious				Bibsonomy			
Partition	HF Resource	LF Resource	HF Tag	LF Tag	HF Resource	LF Resource	HF Tag	LF Tag
Max Freq.	809	224	10598	1097	61	19	151	9
Average Freq.	336.77	42.14	2638.96	74.88	27.11	7.28	35	4.2

Table 6.2: Properties of data partitions

We followed the procedure described in Algorithm 1 to partition tags and URLs based on their frequency distribution to two partitions: low frequency and high frequency. Table 6.2 shows the boundaries for resource and tag frequency in Delicious and Bibsonomy data.

Popular tags and resources are located in the high frequency part of the data and are shown in our figures with “HF”. Tags and resources with less popularity are located in the low frequency part of the data and are shown with “LF” in our charts. It is important to notice the differences between the distribution of tags and resources in the high and low frequency parts in the two data sets. In Delicious, the “HF Tag” has an average frequency of 2638.96 which is far larger than the average frequency of “HF resources”, 336.77. This shows that in Delicious, there are certain popular tags that many users tend to use them. On the other hand, comparison of the averages of different partitions in Bibsonomy shows that the differences between low and high frequency partitions are less extreme.

We show the impact of different attack types on each partition of the data using the local and global metrics. For measuring the global effect we look at the change in the PageRank of the target item in the overall network. So, the global rank includes all users, tags, and resources. The local measures include change in the local rank and hit ratio in each related channel using a particular retrieval algorithm. The results reported here are the final results from averaging several independent runs. In each run, the attack targets are randomly selected from each part of the data (low frequency and high frequency) and the results are recorded.

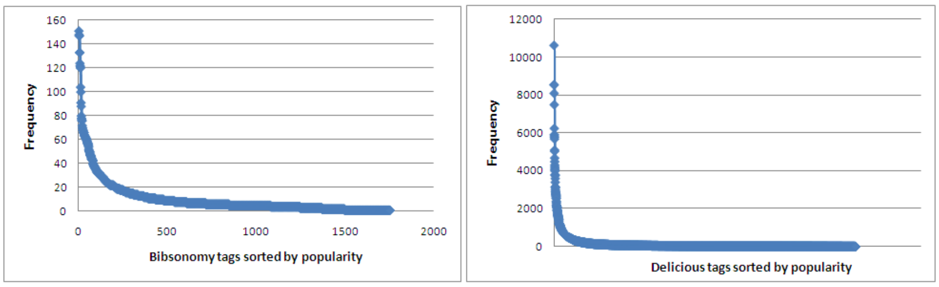


Figure 6.6: Histogram of resource frequency in Bibsonomy and Delicious data sets

6.7.2 Overload Attack

As mentioned in section 6.4.1, the goal of an overload attack is to associate a set of tags with a target resource so that the system retrieves the target resource when one of the tags is given as a query. We implement two variants of this attack: *popular* and *focused*.

Popular Overload Attack

We have two sets of experiments for popular overload attack. In both cases, the target resource is randomly selected from either the high or low frequency partition and the popular tag is randomly selected from the 50 most popular tags in the system. In the first experiment, we choose one single popular tag, associate it with the target resources and test the impact of changing the number of attack profiles. Figure 6.7 shows the global impact of this type of attack for both data sets. We can see the same trend in both data sets, i.e., that increasing the number of attack profiles increases the reciprocal rank for the target resource. As expected, the high frequency resources have higher reciprocal rank compared to the low frequency resources before the attack. The LF partition in Bibsonomy has a higher rate of increase compared to LF in Delicious which suggests that in a smaller tagging system, it is easier for low frequency resources to get higher ranks compared to a popular system such as Delicious.

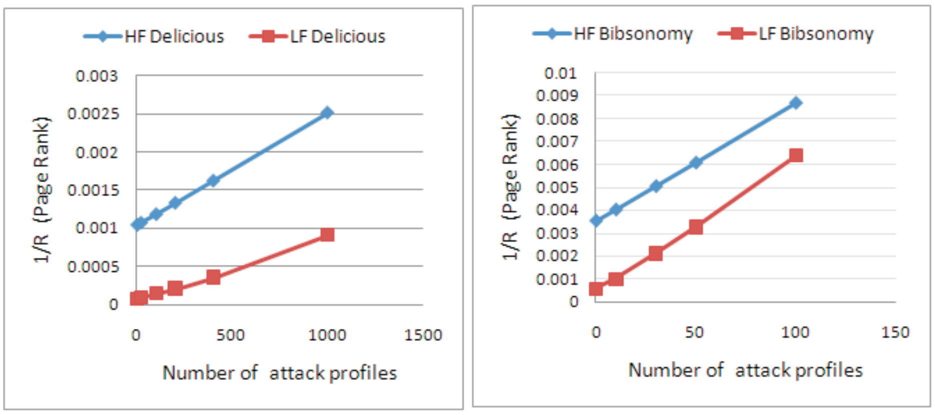


Figure 6.7: Global impact of popular overload attack with one popular tag in each attack profile

Similar effects can be found in local rank improvements, shown in figure 6.8. The results show that much more effort is needed by an attacker to get high ranks for resources located in the low frequency part of the data in Delicious.

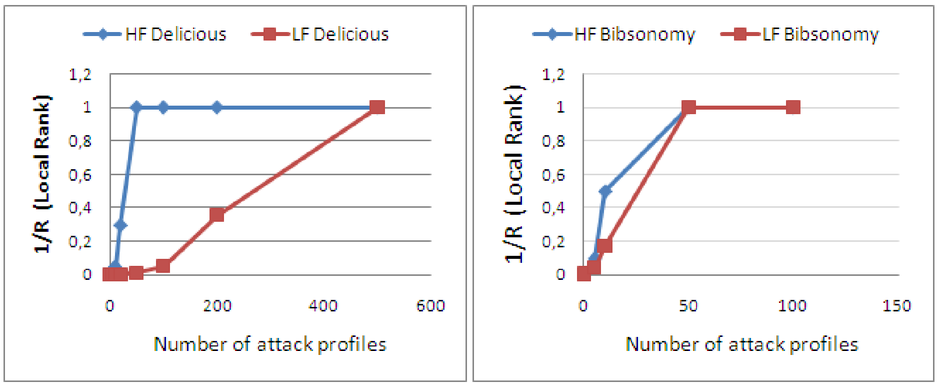


Figure 6.8: Local impact of popular overload attack with one popular tag in each attack profile

Figure 6.9 shows the impact of the attack from the point-of-view of an actual user querying the system; we show the hit ratio of the target for a query consisting of a single popular tag and considering the top 10 results. The results show that a popular overload attack with target resource from LF resources in Delicious has a low chance of success unless the attacker injects large amount of attack profiles. However, such attacks can be successful for the HF resources even for small attack sizes. This difference is due to the fact that most popular resources in Delicious are already associated to popular tags before the attack. So, an small attack can make them visible in the top 10 results. Results from the Bibsonomy data set shows that an attack target from LF and HF part of the data are not so different and the attack can succeed to push the target resource to the top 10 resources by injecting small number of attack profiles (10 injected attack profiles).

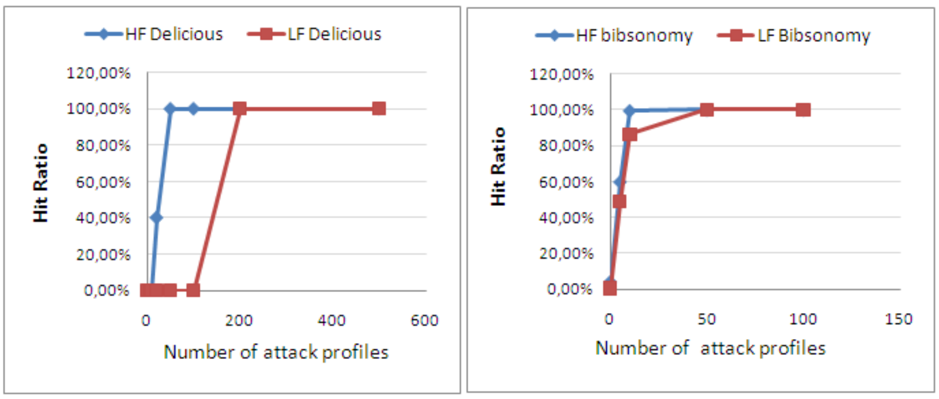


Figure 6.9: Hit Ratio for popular overload attack with one popular tag in each attack profile

For our second round of experiments, we keep the attack size constant and vary the number of popular tags included in each profile. Figure 6.10 depicts the global effect of varying the profile size from 1 to 10 tags. Our PageRank metric allows us to directly compare the effect of attack size vs. profile size. We see that adding more popular tags to each attack profile can be as effective as adding more attack profiles. For example, at attack size 50, associating 2 popular tags to the target resource has more or less the same impact as adding 100 attack profiles with 1 popular tag each. Note that it is easier and less costly for an attacker to associate more popular tags to target resource per profile than it is to add more attack profiles while it might make it more difficult to be detected.

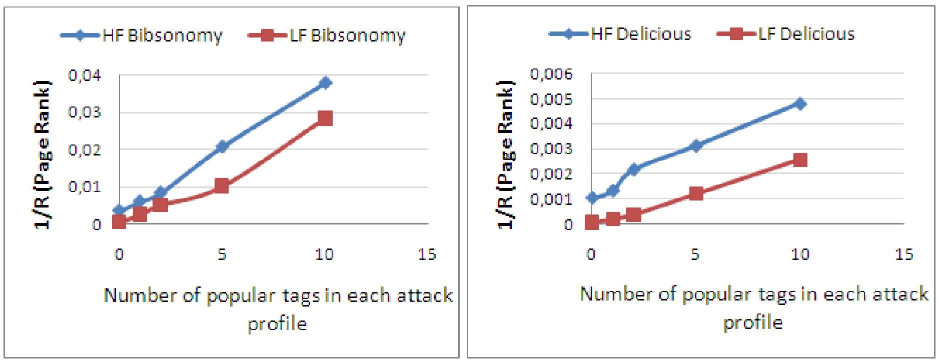


Figure 6.10: Global impact of popular overload attack with n popular tags in each attack profile with 50 attack profiles for Bibsonomy and 200 for Delicious

Focused Overload Attack

In a focused attack, the attacker targets a particular group of users. Such an attack might contain profiles that associate the target URL with specific tags that are interesting to a particular group of users. To measure the global effect of this attack, we bias the preference vector toward the focused tag to see the impact on the system for users who are interested in that specific tag. We use the approach taken in [105] to set the weights for the preference vector. We give higher weights to the focused tag in $\vec{v}$ in equation 6.6. While each user, resource, and tag gets an initial preference weight of 1, the focused tag gets an additional preference weight, $1+|T|$, where $|T|$ is the number of unique tags in the data. The value of damping factor, d, in this case is a representative of the degree to which the target group is focused on a particular topic. We set damping factor $d = .15$, based on the observation that in our test cases, users' most favorite tag forms on average 15% of their profile. We use random tags from the low density partition as our focused tags.

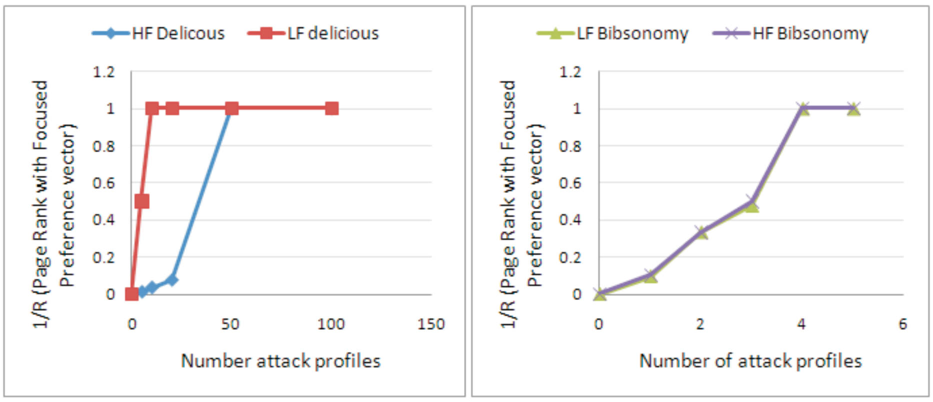
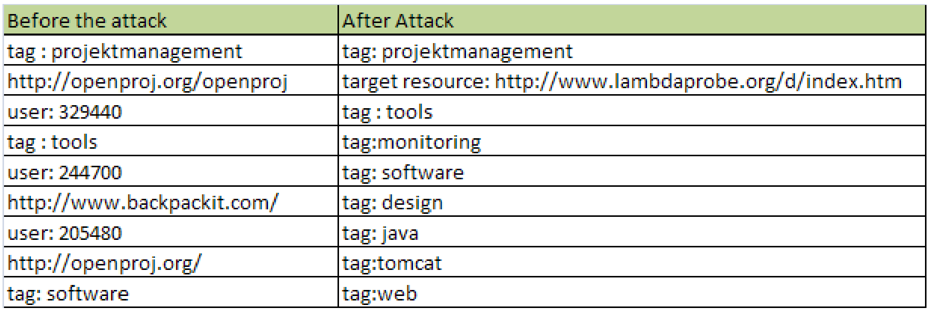


Figure 6.11: Global impact of focused overload attack

The results from both datasets show that the focused overload attack can drastically change the behavior of the system for the target users. As shown in Figure 6.11, even very small attacks can change the rank of the target resource to 1. The hit ratio and local rank improvement using the *tf* retrieval algorithm show similar results, which implies that focused attacks can have an extreme effect on the behavior of the system for a specific target group.



Before the attack	After Attack
tag : projektmanagement	tag: projektmanagement
http://openproj.org/openproj	target resource: http://www.lambdaprobe.org/d/index.htm
user: 329440	tag : tools
tag : tools	tag:monitoring
user: 244700	tag: software
http://www.backpackit.com/	tag: design
user: 205480	tag: java
http://openproj.org/	tag:tomcat
tag: software	tag:web

Figure 6.12: An example of change in the network with focused attack for focused tag "projektmanagement"

In Figure 6.12, we show an example of the impact of a focused attack. In this example our focused tag which is randomly selected from the low frequency partition is "projektmanagement". We can see that before the attack the resource "http://openproj.org/" is the most highly associated resource with this tag. The right side of the table shows the ranking after the attack. The target resource in this attack is "http://www.lambdaprobe.org/d/index.htm" and we can see that not only does the target resource get the highest rank with respect to the focused tag, but also the other resources and tags which are highly associated with the focused tag will change. Tags "monitoring" and "tomcat" happen to get higher rank because they are the tags which are associated with the target resource. As shown in this example, using PageRank can help us understand the impact of attacks in the overall network which was not possible using previous approaches such as hit ratio or hit probability.

6.7.3 Piggyback Attack

To evaluate the Piggyback attack we implement two variations of the basic strategy as discussed before: in the *tag duplication* technique, a certain number of tags are chosen from the popular resource's profile and applied to the target; in the *tag overlap* technique, both the target and the popular resource are tagged a predetermined number of times using tags selected at random.

We perform our experiments using targets from the high and low frequency partitions of the data. In addition to inverse rank based on PageRank, we measure cosine similarity between the target resource and the chosen popular resource to measure the local impact of the attack.

As shown in figure 6.13, the two attack strategies have similar effect at small attack sizes but the tag overlap strategy is relatively more effective at raising the target's PageRank at large attack sizes. One explanation is that in the case of tag duplicate, the attack has the side effect of reinforcing the already-high authorities of the tags in the popular resource's profile. The target only receives a small portion of this increased authority in back-propagation; the popular items' "outflow" is diluted among their many neighbors. In the tag overlap strategy, a more exclusive connection is generated between the popular resource and the target at large attack sizes because the random tags are added to both the target resource and the popular resource.

An interesting comparison is made when looking at the localized metric of cosine shown in figure 6.14. Tag duplicate causes the cosine similarity between target and popular resource to jump considerably after only a small attack, yet it levels out and never surpasses a certain upper bound. Meanwhile, tag overlap requires a greater effort, but appears to grow boundlessly, subject to the size of the attack. The reason for this behavior is that in tag duplicate strategy, the attacker does not change the tag distribution of the popular resource. The cosine similarity is higher when both resources have a similar tag distribution. Thus, the tag overlap strategy can be more effective in large attack sizes to create the same tag distribution for both resources. That, of course, would need more effort from the attacker because the attacker has to create enough fake profiles to change the tag distribution of a popular resource.

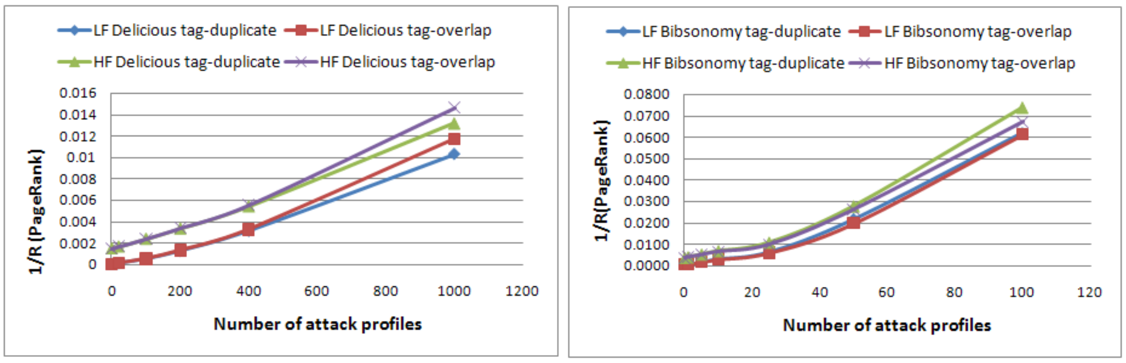


Figure 6.13: Global (PageRank) impact for two variants of the Piggyback attack

Figure 6.15 shows the impact of piggyback attack on hit ratio when using cosine similarity as retrieval algorithm. We can observe that in Delicious data set, it is almost impossible to get high visibility through resource-resource channel when tag duplicate is used as the attack strategy. This result can be explained by looking at the cosine similarity results in figure 6.14. We can observe that after a certain limit, adding more attack profiles does not increase the cosine similarity. In Delicious, there are enough other resources that have a more similar tag distribution to the popular resource. Thus, the attack target can not easily get to the top 10 similar resources. A successful attacker would need to know about the tag distribution of the popular resource and try to mimic such distribution. We can observe that the tag overlap

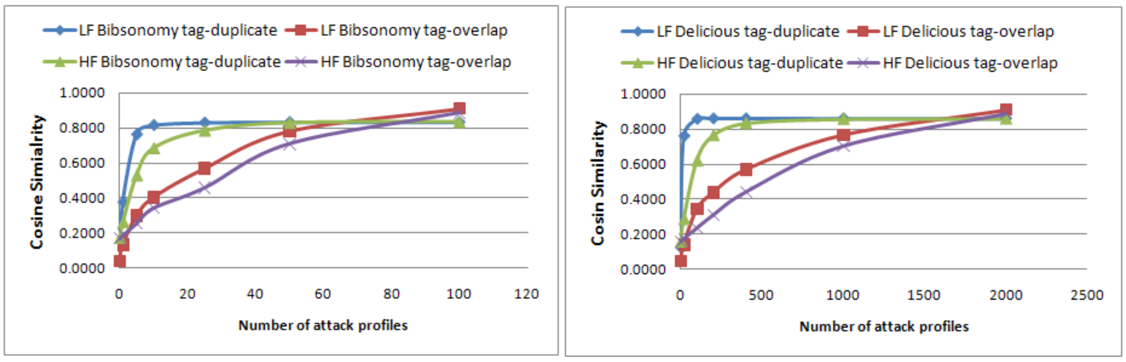


Figure 6.14: Local impact (Cosine similarity) for two variants of the Piggyback attack

strategy needs too much effort and very large attack sizes to be successful in Delicious data set.

In Bibsonomy data set, however, piggyback attack is more successful. Since Bibsonomy is sparser, fewer resources with similar tag distribution exist in the data. Even though the maximum cosine similarly for tag duplicate in both data sets is around .8, we can see that the hit ratio in Bibsonomy gets to 1 after small attack sizes.

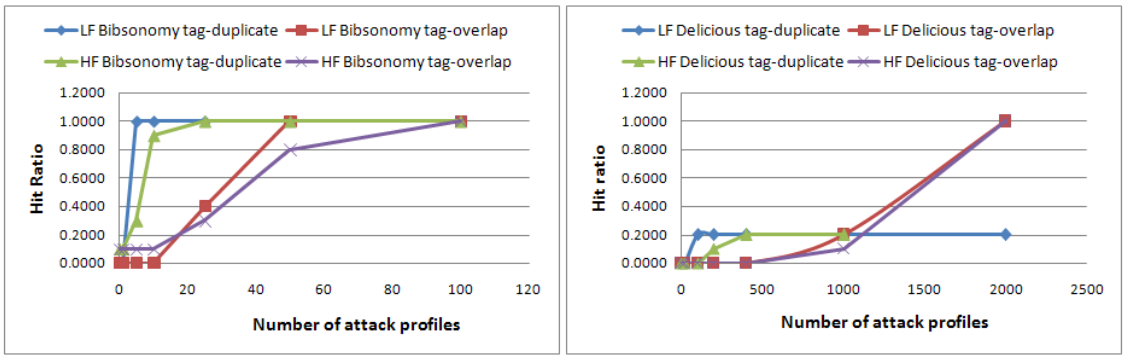


Figure 6.15: Local impact (Hit Ratio) for two variants of the Piggyback attack

6.7.4 Co-occurrence Attack (Tag Push)

The goal of co-occurrence is for a target tag to be correlated with another more popular tag. An attack consists of annotating any resource with both tags, such that they always occur together. The co-occurrence attack profile consists of n popular tags, n resources –randomly selected from the high density partition– and the target tag. Basically, this approach is similar to the *tag overlap* variation of the Piggyback attack, with the difference that in this case the target is a tag and the goal is to associate it with popular tags. We have three sets of experiments for this attack. In all experiments the target tag is selected at random from either partition of the data.

Our first experiment varies the number of attack profiles. An attack profile consists of one popular tag, one random resource, and the target tag. Figure 6.16 shows the change in the overall rank of the target tag. Looking at the the results from Delicious data set, we can see that there is a huge difference between the overall ranks of HF and LF tags even before the attack (attack size 0). The HF tags have a much higher rank on average than the LF tags. We can observe that the attack is not as successful when the target is selected from the low frequency tags. This is due to the fact that the popular tags in delicious are well established and it is not easy for an attacker to generate a high rank for some random tag in the overall network. Tags that are located in the HF part have already considerable high rank before the attack, they are well connected to popular items in the system and they can easily get higher rank with small attacks.

In Bibsonomy data set, however, we can see much less difference between the low and high frequency before the attack. Thus, it is easier to push low frequency tags compared to Delicious. In both data sets, as expected, high frequency targets are more successful.

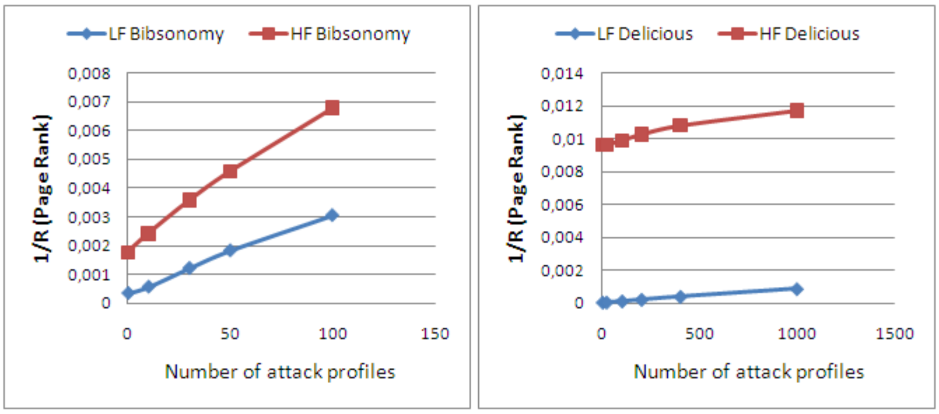


Figure 6.16: Global impact of co-occurrence attack with 1 popular tag and 1 resource in each attack profile

The local results reported in figure 6.17 show how the co-occurrence attack can be effective in the context of “related tags”. We use cosine similarity to find similar tags to the popular tag. Figure 6.17 shows that even small attacks can change the related tag rank in favor of the target tag. In Bibsonomy, for example, adding 10 attack profiles makes the target tag most similar to the popular tag.

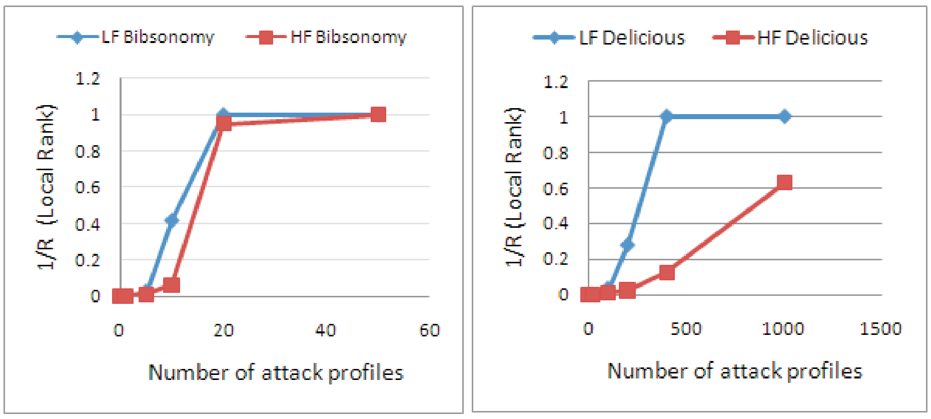


Figure 6.17: Local impact of co-occurrence attack with 1 popular tag and 1 resource in each attack profile

Our next experiment is to change the number of popular tags in each profile while keeping the number of resources and attack size fixed. In this experiment, each attack profile consists of *n* popular tags, one random resource and the target tag. The left side of figure 6.18 shows the results for this type of attack. As we can see, there is no correlation between the number of popular tags and the rank of the target tag. This result is to be expected since changing the number of popular tags in one attack profile will not help the target tag get a higher rank, but it will help the associated resource to get a higher rank as the target tag occurs only once in each attack profile.

Our third experiment changes the number of resources to be tagged. Thus, each attack profile consists of one popular tag, *n* resources, and the target tag. The results for this experiment can be seen in the right side of figure 6.18. We see that the number of resources can considerably effect the rank of the target tag. This means that one attacker can associate a target tag with a popular tag over many resources and easily get a high rank for the target tag.

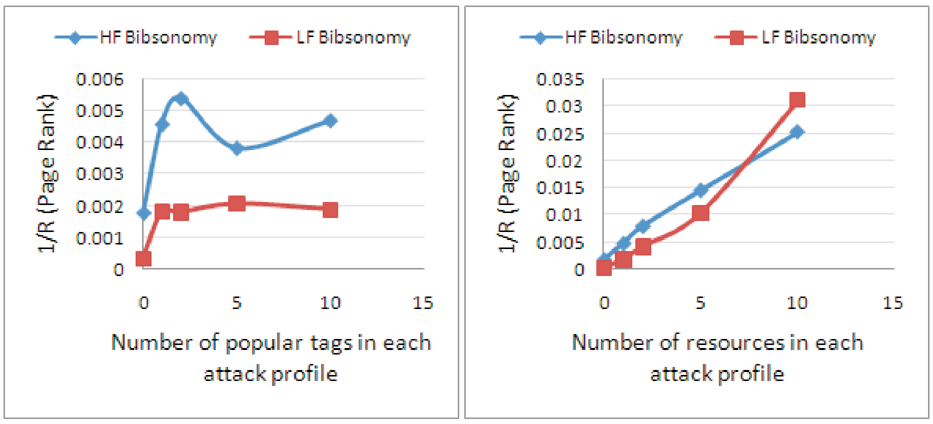


Figure 6.18: Global impact of co-occurrence attack with n popular tag (on the left) and n resources (on the right) in each attack profile with 50 attack profiles injected to Bibsonomy data set

6.7.5 Comparison Of Attack Impact On Different Data Sets

Figures 6.19, 6.20, and 6.21 show the comparison of the global rank improvement of the attack target in different data sets. The attack size is the percentage of the attack profiles to the total number of users in each data. We can see that in all different attack types, Bibsonomy is more vulnerable to attack than Delicious. This is because Bibsonomy is a much sparser data set than Delicious; thus, it is easier to change the behavior of the system by injecting attack profiles.

We can observe a bigger gap between the impact of overload attack on the low frequency resources. This gap confirms our previous observation that pushing low frequency resources in Delicious needs huge effort while it is easier to push such resources in Bibsonomy by an overload attack strategy.

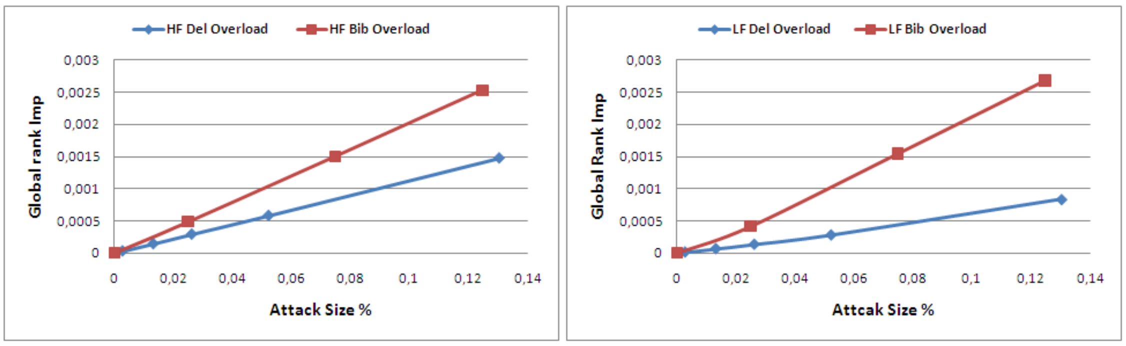


Figure 6.19: Comparison of the global impact of overload attack on different data sets

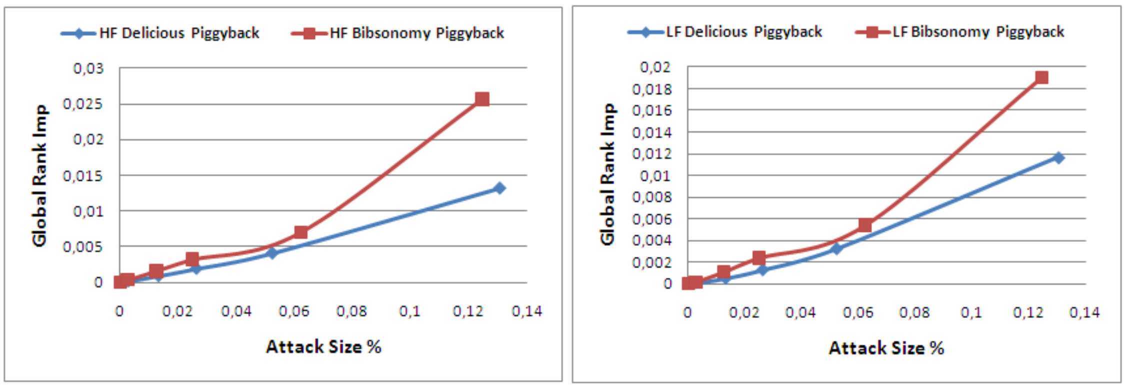


Figure 6.20: Comparison of the global impact of piggyback attack on different data sets

6.7.6 Comparison of Different Attack Types

Figures 6.22 and 6.23 show the comparison of the global rank improvement of the attack target for different attack types. Each chart shows the three types of attack- popular overload, piggyback with overlap strategy, and co-occurrence- on each partition of the data. We can have several important observation by analyzing these charts.

Figure 6.22 shows the results for the Bibsonomy data set. We can see that there is not a significant difference among the different attack types when the target of the attack is selected from high frequency part of the data while there is considerable difference for targets from low frequency part of the data. The overload attack has the highest impact on rank improvement of low frequency targets followed by piggyback and co-occurrence attacks .

Looking at figure 6.23, we observe a different behavior in Delicious data set. There is not a significant difference among different attack types when the attack target is selected

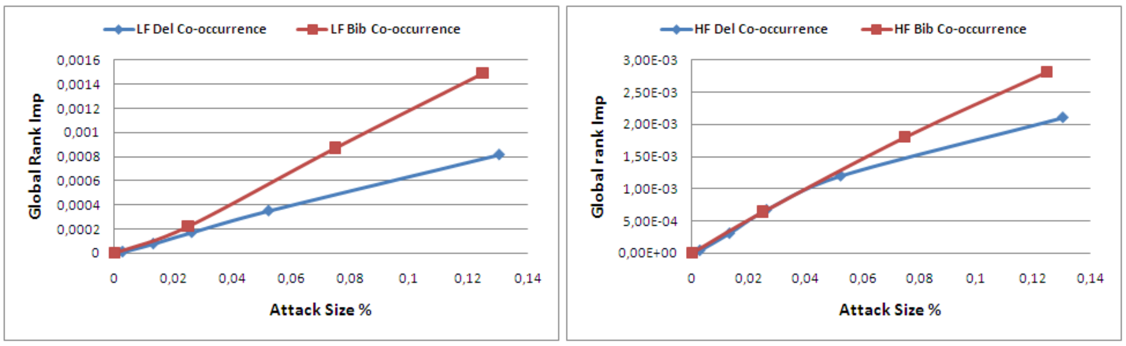


Figure 6.21: Comparison of the global impact of co-occurrence attack on different data sets

from the low frequency partition of the Delicious data. However, for attack targets from the high frequency partition, we can see that the co-occurrence attack has a higher impact than the other two attack types.

To explain these differences, we look back to the data distribution table 6.2. Comparison of the average and maximum frequency of tags and resources in Bibsonomy and Delicious data sets show that the low frequency resources in Bibsonomy are more often used than the low frequency tags (average frequency of 4.2 for LF tags compared to average frequency of 7.28 for LF resources) while in Delicious the LF tags have a higher popularity than the LF resources (average frequency of 74.88 for LF tags compared to average frequency of 42.14 for LF resources). This difference in the data distribution is because of different kind of users in each system, their tagging behavior, and the kind of resources tagged in each system. Bibsonomy is more focused on scientific articles while Delicious covers more general topics. In Bibsonomy data set the LF resources are more vulnerable to attack which makes the overload and piggyback attacks more successful than the co-occurrence attack.

Table 6.2 shows that in Delicious data “HF tags” are significantly more popular than HF resources(average frequency of 2638.96 for HF tags compared to 336.77 for HF resources). Thus, attacks with targets from HF tags can be very effective to achieve a higher rank. Therefore, co-occurrence attack is the most successful attack on HF targets in Delicious.

6.7.7 Comparison of Local Impact

Finally, we compare the local impact of different attack types. We compare the hit ratio of different attacks on different data sets. In figure 6.24, the size of the attack shows the percentage of the injected profiles to the number of users in each data set. The results of this experiment can provide us with several valuable observations. First, we can see that even though co-occurrence attack with HF tag targets in Delicious can impact the targets to

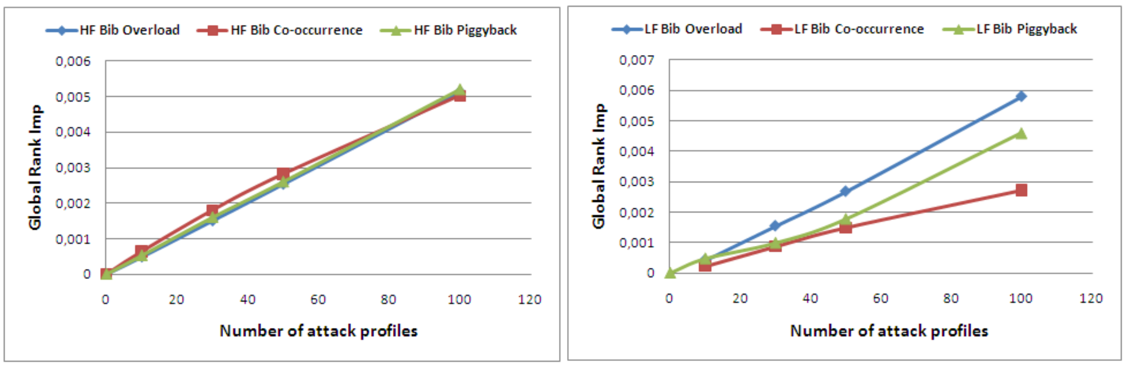


Figure 6.22: Comparison of the global impact of different attack types in Bibsonomy data set

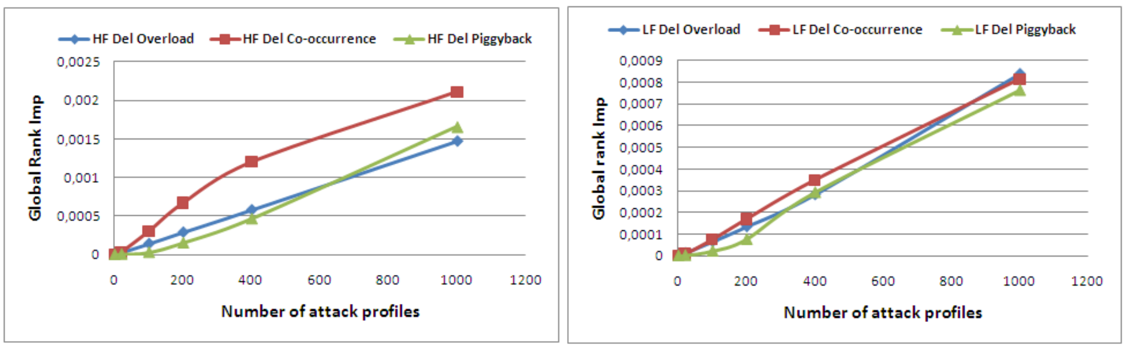


Figure 6.23: Comparison of the global impact of different attack types in Delicious data set

get higher rank, it is not as successful to change the output of the system in the related tag navigation channel. This is due to the fact that the HF tags in Delicious are well established and they are strongly connected to the existing folksonomy graph. Thus, there are existing related tags that are already connected to these tags through many other resources.

On the other hand, we can observe that overload attack can change the system output with relatively small attacks. While Delicious shows to be more robust than Bibsonomy for piggyback and co-occurrence attacks, it is more vulnerable against overload attack for target resources selected from HF partition. Of course, the robustness of each system in each navigation channel is relevant to the retrieval algorithms. We have used cosine similarity for the tag-tag and resource-resource channels and tag frequency, *tf*, for the tag-resource channel. In general, we can see that cosine similarity is more robust than *tf* for a denser data set.

Based on our hit ratio results, we can conclude that tagging systems can tolerate attack sizes up to 1% of their total number of users before their performance significantly degrade. This number can decrease considerably if the attack profiles are designed in a way that they generate more associations in the folksonomy. For example, profiles with larger number of popular tags in overload attack or larger number of resources in co-occurrence attack. Our results from the focused attack shows that this tolerance significantly decreases when attacker targets a particular set of users. For example, for focused overload attack the system can tolerate only up to .1% attack profiles before the results are totally defected; which suggests that very small attacks can change the the behavior of the system.

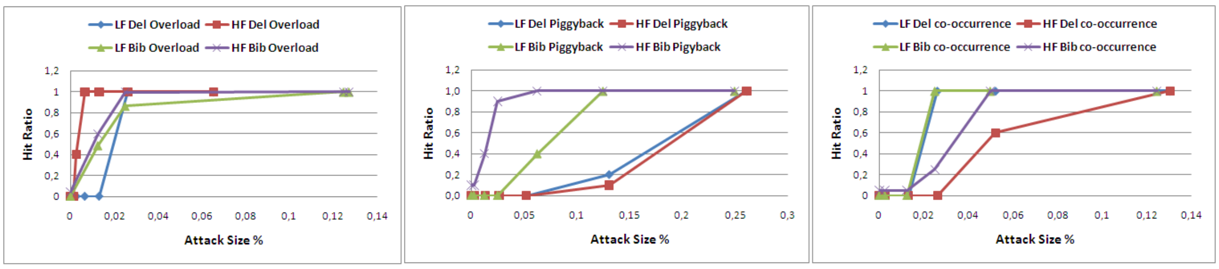


Figure 6.24: Comparison of the local impact of different attack types in different data set

6.8 Discussion

Our results show that tagging systems can be quite vulnerable to attack. Comparison of the local and global impacts of different attack strategies reveals which attack types are more successful on different systems.

Both piggyback and overload attack are used to push a target resource into the system. Our results show that large systems such as Delicious are more vulnerable to overload attack than to the piggyback attack. Piggyback attack needs more knowledge about the distribution of the data in the system or very large attack sizes to impact the system while overload attack can push the target resource with only having knowledge about the popular tags in the system. Our results on the global impact of overload attack shows that adding more number of popular tags in one attack profile can be as successful as adding more attack profiles.

The results concerning focused overload attacks showed that a goal-oriented attack which targets a specific user group can be easily injected into the system. While producing this attack does not require a great deal of effort or knowledge from an attackers perspective, it may be more difficult to detect this kind of attack since it resembles the natural behavior of a person who is interested in a specific topic.

Results from the co-occupance attack show that systems that provide related tag can be vulnerable to tag push. Our results on the local level show that this kind of attack needs large number of attack profiles to be successful in large tagging systems. However, our results also show that increasing the number of resources in each attack profile, can be as successful as increasing the number of injected attack profiles. While it is easier for an attacker to add more resources in one attack profile than to inject more attack profiles, it might make it more difficult to detect since such profiles are more similar to an actual user profile.

6.9 Conclusion and Future Work

In this chapter, we discussed the problem of security and robustness in social tagging systems. We introduced a framework to model the navigation channels in social tagging systems and we identified different types of potential attacks against the system through different navigation channels. We introduced an effective measurement, based on PageRank, to evaluate the global impact of attacks against social tagging systems. We modeled three attack types: Overload (popular and focused), Piggyback, and Co-occurrence, experimenting with real-world datasets. We showed local and global impacts of each attack type on each data set. Our results help the social tagging system administrators to have a clear understanding about potential attacks against their system, how an attack profile might look like and what attack targets are more vulnerable.

There is a lot of opportunity for further investigation in this area. Future work can include modeling other attack types and compare their impacts on the system. Taking into account other important points including the network dynamics and temporal impacts can help understand the way attacks propagate into the system and how they can be distinguished from normal users. Developing more robust algorithms as well as approaches for detection and prevention of attacks can be considered as areas for further research. To better understand this problem and find approaches for solving the problem, we propose several important points that can direct future researchers for advancing in this area.

Network Dynamics Every annotation added to the system changes the structure of the network. Some of these changes will have greater impact for user experience than others, for example, by connecting previously unrelated parts of the network together. We used PageRank algorithm to measure the changes in the global rank of the attack target. However, the impact of different kind of attacks on the network dynamics can be studied in more depth. We can find the aggregated rank improvement to measure the impact of attack not only on the attack target but also on all other element of the folksonomy network. The aggregated rank improvement can be calculated as the sum over all rank changes in the network:

$$AGImp = \sum_{i=1}^{N} |Imp_i| \tag{6.7}$$

Where $N = |U| + |R| + |T|$ is the total number of graph vertices (sum of number of all users, resources, and tags) and Imp_i is the rank improvement of each item after the attack. Other approaches can be studied based on network theory. Studies of the patterns and structures of propagation in large networks such as in [120] can be applied to discover how the propagation of attacks differ from the natural network evolution.

Temporal Impacts Time is a very important factor in the system evolution and should be considered in the attack models, retrieval algorithms (recency-based algorithms), and evolution of the system. Considering the temporal aspects of the attacks is an important issue for both studying the impact of attack in longer period of time and for detection of attacks.

Attack Detection A social tagging system takes in a stream of data, and the problem is to detect trends in that data that may indicate an attack before the attack is significant enough to impact users. Our work in this chapter can suggest how an attack profile might look like and what are the features that can be considered for machine learning algorithms for attack detection. In addition, the temporal network evolution properties can be used for attack detection. Studying the impact of attacks over time and comparing them with natural evolution of the network can be a clue to detect the attack trends. The temporal sequence of tag application is also an important clue that can help distinguish attackers.

Attack Prevention Considering the users of a social tagging application as a society, they should be able to protect themselves if they have enough resources to manage the system. Our hypothesis is that there are enough trustful users in the system that if they would have the ability to influence the system, they would be able to protect the system from attacks or spam. Trust-based systems allow for combining classical visibility measures with trust-based recommendations, giving enhanced visibility measures that can help prevent the influence of an attack to the system.

6.10 Acknowledgment

The work in this chapter was supported by the National Science Foundation Cyber Trust program under Grant IIS-0430303 and a grant from the Department of Education, Graduate Assistance in the Area of National Need, P200A070536. Thanks to J. J. Sandvig, Tom

Schimoler, Jonathan Gemmell, Runa Bhaumik, Bamshad Mobasher and Robin Burke for their considerable contributions in this work.

Chapter 7

Conclusion

Social Web applications have emerged as powerful applications for Internet users allowing them to freely contribute to the Web content, organize and share information, and utilize the collective knowledge of others for discovering new topics, resources and new friends. The social Web applications realize the insights of many users rather than a few "experts", making the information on the Websites more dynamic, and better able to serve niche communities as well as general audiences.

While social Web applications such as social tagging systems have many benefits, they also present several challenges due to their open and adaptive nature. The amount of user-generated data can be extremely large and since there is not any controlled vocabulary or hierarchy, it can be very difficult for users to find the information that is of their interest. In addition, attackers may attempt to distort the system's adaptive behavior by inserting erroneous or misleading annotations, thus altering the way in which information is presented to legitimate users.

In this thesis, we presented data mining and machine learning approaches to address these problems in two different directions. First, we focused on the role of recommender systems to aid the user in contributing to the system. Second, we introduced a framework to model potential attacks against social tagging systems and evaluated their impact on the system.

In this chapter, we provide a summary of our research in this thesis. First, we briefly review the answers to the research questions we had presented in the beginning of the thesis. Next, we present the main contributions of our work and finally we provide several directions for further research in this area.

7.1 Answer to Research Questions

Based on our motivations and objectives, we formalized several research questions which this thesis has focused on. In this section we summarize the answers to those research questions.

7.1.1 Objective 1: Matching Recommendation Technology and Domains

We started the thesis with a broad survey on recommender systems with the goal of assisting researchers and developers in selection and application of these systems. The main research questions that we addressed are the following.

- What are the main characteristics of a domain that influence the choice of selecting the appropriate recommendation technology?

We identified six domain characteristics that impact the choice of selecting the appropriate recommendation technology. These properties are heterogeneity, risk, churn, interaction style, preference stability, and inscrutability.

A **heterogeneous** domain encompasses many items with different characteristics and different goals they can satisfy. **Risk** is the degree of uncertainty that a user can tolerate in accepting a recommendation. **Churn** indicates the life span of the items in the domain. **Interaction style** characterizes the kind of interaction with the system. **Preference stability** refers to the life time of user preferences and **inscrutability** indicates the degree of explanation needed for a recommendation.

- What are the main knowledge sources in recommendation systems and how do they relate to the recommendation technology?

We identified three types of knowledge sources in recommender systems: (1) **Individual knowledge** which is knowledge about the target user (2) **Social knowledge** which is knowledge about the larger community of users other than the target user (3) **Content knowledge** which is knowledge about the items being recommended and/or the domain of recommendation, including how recommended items are used and what needs they satisfy. We further categorized individual knowledge to four categories of opinion, demographic, behavior, and requirement. In content knowledge category, we distinguished between item features and domain knowledge.

Next, we mapped the knowledge types to recommendation technologies. Social knowledge naturally maps to collaborative filtering, item features map to content-based recommendation, and domain knowledge is appropriate for knowledge-based techniques. While, the choice of kind of individual knowledge is dependent on the interaction style, we can generally map opinion, behavior, and demographics to collaborative filtering and content-based techniques while requirements (query, preferences, constraints, context) depending on their type can map to content-based and knowledge-based techniques.

- How can we map the domain characteristics to recommendation technologies?

We mapped the domain characteristics to recommendation technologies based on the knowledge sources that each technology requires. In a nutshell, high-risk, scrutability, and preference instability generally lead to knowledge-based recommendation. Heterogeneous domains with implicit input are handled largely with collaborative recommendation and high churn domains go along with content-based techniques.

7.1.2 Objective 2: Improving Link Analysis for Tag Recommendation in Social Tagging Systems

We focused on graph-based recommendation approaches in folksonomies with the goal of improving the quality of personalized tag recommendation. We focused on answering the following questions.

- How can we model the folksonomy as a graph so that we can capture the flow of information?

We modeled the folksonomy as a weighted directed tripartite graph. The vertices of the this graph are users, resources, and tags. The edges are bidirectional but the weight of each direction is different from the weight of the other direction. The weights are defined based on the fact that the user navigation from one object (user, resource, or tag) to another object in a folksonomy is not symmetric and by considering different weights on the edges of each direction we can better model the navigating from one node to the other. We consider higher weights for in-links to more popular nodes.

- How can we use the model for tag recommendation?

We used an adaptation of the PageRank algorithm introduced by [102] for ranking the nodes in the folksonomy. For personalized tag recommendation, we use personalized PageRank algorithms where we set the preference vector in a way that it has high emphasis on the target user and resource.

- Does the proposed model produce better recommendation than the previous approaches?

To evaluate our approach, we used three real world data sets from Delicious, Bibsonomy, and Citeulike. We used common evaluation methodologies for tag recommendation and used precision and recall as evaluation metrics.

We suggested two improvement to the existing graph-based tag recommendation approach, FolkRank. First, we investigated the influence of the parameters for tag recommendation and experimentally showed that with correct parameterizations Adapted PageRank can outperform FolkRank. Second, we showed that with modeling the folksonomy as a directed weighted graph, we can get even more improvements.

7.1.3 Objective 3: Using Recommender Systems for Continuous Ontology Development

Our third objective in this thesis was to investigate how recommender systems can help users to collaboratively develop an ontology. In particular, we were interested in the following questions.

- How can we aggregate user activities in a social tagging environment to infer semantic similarity?

We suggested contextual cues to calculate the similarity between the concepts. Users enter new concepts in a specific context; for example, they tag a specific document. We modeled each concept as a vector of the items related to it and used cosine similarity to find the contextual similarity among concepts.

- How can we use machine learning to learn from existing semantic relations in a concept hierarchy to predict semantic relations between a new concept and existing concepts?

We developed a learning algorithm that measures the similarity between a new concept and existing concepts and based on the location of the most similar concepts, suggests a place for the new concept. We introduced sub-super affinity (SSA) based on distances in a hierarchy and developed an algorithm that can predict the SSA value between each new concept and all existing concepts. We evaluated our algorithm using a data set from Wikipedia and our evaluation shows excellent results. We could get up to 90% accuracy by setting the threshold high enough. In addition, we performed an expert evaluation which confirms our algorithm can produce results which are as accurate as Wikipedia's existing hierarchy.

- How can the algorithm support collaborative ontology development?

We suggested to apply our machine leaning algorithm in a Web 2.0 environment to support continuous ontology development at the time of their use. From this perspective, ontology is the backbone of a social semantic system that can be used by users for structuring and sharing information items. Our algorithm suggests semantic relations between the existing concepts and the new concepts that users enter into the system and helps them collaboratively and continuously develop the ontology.

7.1.4 Objective 4: Combating Attacks Against Social Tagging Systems

Our final goal in this thesis was to explore how data mining can help to combat attacks against social tagging systems. Our study in this topic led us to answer the following questions.

- How can we systematically study the problem of attacks against social tagging systems?

We presented the dimensions that characterize an attack and outlined a framework to model the navigation channels in social tagging systems. We identified different types of potential attacks against the system through different navigation channels.

- How can we evaluate the impact of attacks in social tagging systems?

We introduced two different kinds of evaluation metrics to measure the impact of attacks. First, local metrics which are dependent on the retrieval algorithms and can be different based on the navigation channel. Local metrics are useful for evaluating how end users are affected by an attack. Second, we introduced global metrics, which measure the impact of attacks globally and can observe changes in the underlying network of folksonomy. Global metrics allow us to study attack effectiveness in a more holistic manner.

- What model of attacks are more successful?

We performed extensive experimental study on two real world data sets, Delicious and Bibsonomy, and studied the local and global impact of three types of attack. Our results show that focused attacks in which the attacker targets a particular set of users can be extremely successful to change the behavior of the system for the target users. While these type of attacks do not need so much effort from an attacker perspective, they are more difficult to detect from the system perspective.

- How many malicious users can a tagging system tolerate before results significantly degrade?

We recorded the hit ratio results for different kinds of attacks to monitor how the attack influences the output of the system from a user perspective. Our results show that, in general, social tagging systems can tolerate attack sizes up to 1% of their total number of users before their performance significantly degrade. This number can decrease considerably if the attack profiles are designed in a way that they generate more associations in the folksonomy. For example, profiles with larger number of popular tags in overload attack or larger number of resources in co-occurrence attack.

Our results from the focused attack shows that the tolerance significantly decreases when attacker targets a particular set of users. For example, for overload attack, the system can tolerate only up to .1% attack profiles before the results are totally infected.

- How much effort and knowledge is needed by an attacker to attack the system?

Our results show that most attack types do not require a great deal of knowledge about the details of the tagging systems. Aside from one type of piggyback attack (tag duplicate) in which attackers need to have knowledge about the tag distribution of a resource in order to have more success, other attack types do not need more information than publicly available information on the Website. In terms of effort, our results show that some attacks need larger attack sizes to change the output of the system. For example, for pushing a target resource, overload attack needs less effort than piggyback attack.

7.2 Summary of Contributions

This thesis presents a combination of (a) conceptual work (b) design and development of data mining and machine learning algorithms (c) empirical work, measurements and experiments. This naturally closes the loop between design and modeling on one hand, and empirical measurement and evaluation on the other hand.

7.2.1 Conceptual Contributions

This thesis presents two major conceptual contributions to the field: (1) Matching domains and recommendation technologies (2) A framework for analyzing impact of attacks against social tagging systems

Matching domains and recommendation technologies

Our in-depth study in the recommender systems literature led us to a valuable contribution in the field to guide academics and implementers to select and apply recommendation technologies based on the domain properties. We categorized recommendation technologies based on their knowledge sources, identified domain characteristics that influence on selecting the appropriate recommendation technology and mapped domain characteristic to the proper knowledge source and technology.

Attack framework

In this thesis, we introduced a systematic approach to study the the problem of security and robustness in social tagging systems. We introduced a framework to model the navigation channels in social tagging systems. We analyzed different types of potential attacks against the system through different navigation channels. In addition, we introduced approaches to evaluate local and global impact of attacks.

7.2.2 Algorithm Development

We designed and developed several new data mining and machine learning algorithms in this thesis for different purposes: personalized tag recommendation in folksonomies and recommendation of semantic relations.

Personalized tag recommendation

We presented a weighted directed graph of folksonomy that can capture the flow of information in a folksonomy. We used our proposed graph model and applied an adaptation of PageRank algorithm for tag recommendation.

Recommendation of semantic relations

We presented a recommendation algorithm to support continuous ontology development in a Web 2.0 environment. Our algorithm learns from an existing concept hierarchy and suggests a place for a given new concept. The algorithm uses the contextual and string similarity between the new concept and existing concepts and based on the place of the most similar concepts, recommends potential super-concepts for the new concept. Our work is a novel approach to ontology learning since unlike previous approaches, we do not focus on creating an ontology in one off effort. Instead, we propose an approach to learn from an existing hierarchy and to assist users to further develop the hierarchy at the time of their use.

7.2.3 Empirical Contributions

We applied our algorithms on several real world data sets and our extensive experiments and evaluations led us to several important observations as well as evaluation of our developed algorithms.

Evaluation of the algorithms

We evaluated our tag recommendation algorithm by extensive experiments on three real world data sets: Delicious, Citeulike, and Bibsonomy. Our evaluation results show that our approach improves previous successful tag recommendation algorithms.

We evaluated our algorithm for recommendation of semantic relations on Wikipedia category hierarchy, and in part on the Floyd data set from SAP. Our evaluation results show the algorithm provides good quality recommendations.

Observations

We carried out comprehensive experiments to observe the impact of different types of attack against social tagging systems and we had several important observations. We provide a summary of our observations here.

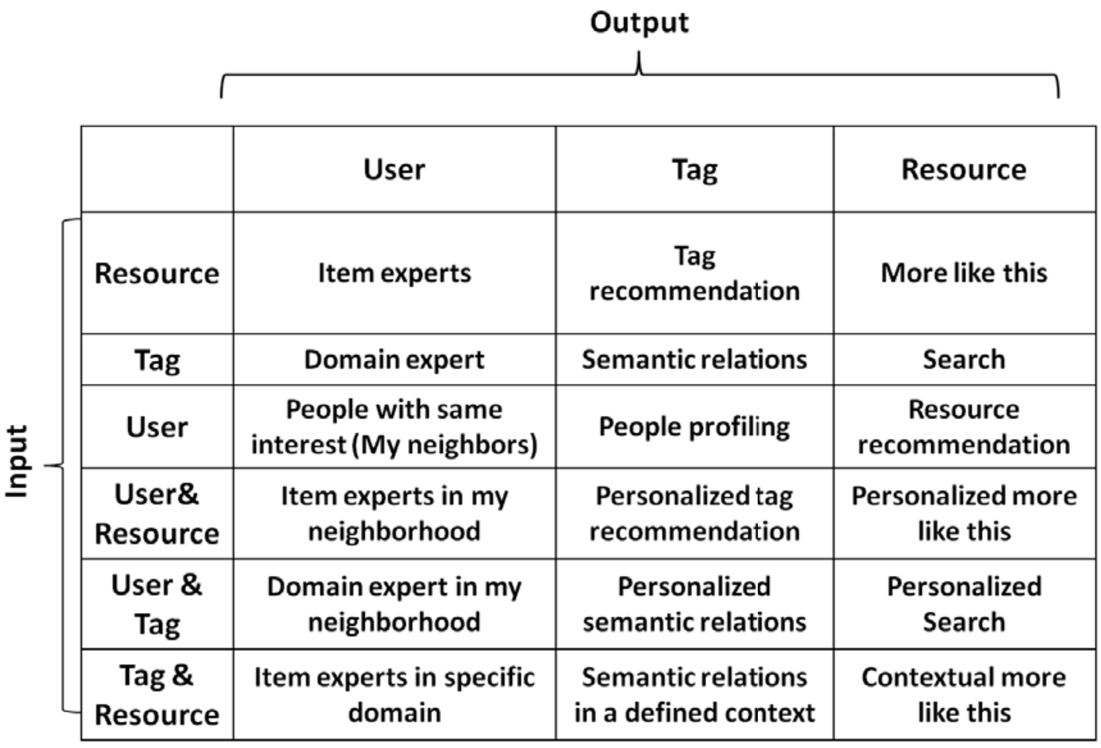
- Smaller tagging systems with sparser folksonomy relations among tags, resources and users are generally more vulnerable to attacks than larger systems with denser relations.
- Attack targets from the high frequency partition of the data are more successful in changing the global network of the tagging system compared to targets from low frequency partitions.
- The vulnerability of different navigation channels in a tagging system highly depends on the distribution of the data in the system, which naturally depends on the type of tagging system and the tagging behavior of the users. In general, we could observe that associating the target element of the attack to popular tags is more successful than associating them to popular resources (Overload attack is more successful than piggyback).
- The most successful attacks are focused attacks which target a particular group of users. These attacks do not need much effort or knowledge about the system but detecting them can be a big challenge.

7.3 Future Directions

The long-term goal of this research is to support users of social Web applications for three purposes. First, to reduce their effort to bring in meaningful contributions. Second, to assist them to efficiently navigate through the system and easily find items of interest, and third, to provide them the security that the information is trustworthy even though the system is open to the public.

In our thesis, we made several steps towards this long-term goal. We suggested techniques for tag recommendation and semantic relation recommendation. We studied the local and global impact of attacks into the system and presented a comprehensive framework which can provide directions for future research in this area. There are, however, plenty of directions to further explore this area.

7.3.1 Explore Other Recommendation Tasks


Output
Input

	User	Tag	Resource
Resource	Item experts	Tag recommendation	More like this
Tag	Domain expert	Semantic relations	Search
User	People with same interest (My neighbors)	People profiling	Resource recommendation
User& Resource	Item experts in my neighborhood	Personalized tag recommendation	Personalized more like this
User & Tag	Domain expert in my neighborhood	Personalized semantic relations	Personalized Search
Tag & Resource	Item experts in specific domain	Semantic relations in a defined context	Contextual more like this

Figure 7.1: An overview of possible recommendation tasks in social tagging systems

We introduced the different navigation channels in social tagging systems in chapter 6. Considering all the possibilities to navigate in the system, many research directions can be defined. An overview of possible research tasks on social tagging systems is presented in figure 7.1. This table is an extension of the work presented by Clements [51]. Each task is in the form of selecting one or two object types as input and then finding other related objects as the output.

In this thesis, we have contributed in two tasks: personalized tag recommendation and semantic relations. Related work on social tagging has focused on personalized tag recommendation, personalized search, and personalized item recommendation. Other interesting and useful tasks still remain largely unexplored, such as finding like-minded users (my neighbors or domain experts in my neighborhood), and selecting specific items to get recommendations for similar ones (More like this).

Considering users' needs and their motivations of using the system are important considerations to select which research tasks to explore.

7.3.2 Network Evolution and Attacks

Aside from variety of potential research directions in the area of recommendation in social tagging system, there is a lot of opportunity for interesting research in extending our work on combating attacks against social tagging system. We provided several ideas in section 6.9 for further analysis on attacks against social tagging system. Here, we provide a more long-term vision on this area.

One important question in social media is how does the information propagate? How do people influence the propagation of the information? How does a topic get a boost in the network? Answers to these questions can help us to detect the trends in the social network and build predictive models of upcoming trends. In particular, it can help us detect attacks which are designed to create a boost for a specific target.

In addition, mechanisms that can help a network to evolve in a healthy manner and naturally resist attacks can be investigated. For example, it is valuable to study which nodes and activities have more influence on the global structure of the social network. For this goal, it is important to study how local behaviors propagate to global scale and what factors make a specific node more influential than others.

7.3.3 Scalability And Real-time Analysis

Although we have used real world data sets from popular social tagging systems in our research, it is still just a small part of all the data available in these systems. It is valuable to research on scalability of the approaches on large scale data and design models that are able to perform predictive analysis on massive amount of data to find patterns that are practically unobservable at smaller scales.

In addition, our analysis is done off-line without real user interactions. Evaluation of recommendation algorithms would be more valuable if they can be done in real-time setting with actual users. Ideally, such experiments would have to be run in cooperation with one or more popular social tagging sites to attract a large enough group of test subjects to be able to draw statistically significant conclusions about the differences in performance.

Bibliography

[1] "Weighted pagerank algorithm," in *CNSR '04: Proceedings of the Second Annual Conference on Communication Networks and Services Research*. IEEE Computer Society, 2004, pp. 305–314.

[2] K. Aberer, P. Cudré-Mauroux, A. M. Ouksel, T. Catarci, M.-S. Hacid, A. Illarramendi, V. Kashyap, M. Mecella, E. Mena, E. J. Neuhold, O. D. Troyer, T. Risse, M. Scannapieco, F. Saltor, L. D. Santis, S. Spaccapietra, S. Staab, and R. Studer, "Emergent semantics principles and issues." in *Proceedings of the 9th International Conference on Database Systems for Advanced Applications (DASFAA'04)*, ser. Lecture Notes in Computer Science, Y.-J. Lee, J. Li, K.-Y. Whang, and D. Lee, Eds. Springer, 2004, pp. 25–38.

[3] S. Aciar, D. Zhang, S. Simoff, and J. Debenham, "Informed recommender agent: Utilizing consumer product reviews through text mining," in *WI-IATW '06: Proceedings of the 2006 IEEE/WIC/ACM international conference on Web Intelligence and Intelligent Agent Technology*. Hong Kong: IEEE Computer Society, 2006, pp. 37–40.

[4] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM Trans. Inf. Syst.*, vol. 23, no. 1, pp. 103–145, 2005.

[5] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.

[6] B. Adrian, L. Sauermann, and T. Roth-Berghofer, "Contag: A semantic tag recommendation system," in *Proceedings of I-Semantics' 07*, T. Pellegrini and S. Schaffert, Eds. JUCS, 2007, pp. pp. 297–304. [Online]. Available: http://www.dfki.uni-kl.de/~sauermann/papers/horak+2007a.pdf

[7] C. C. Aggarwal, J. L. Wolf, K. lung Wu, and P. S. Yu, “Horting hatches an egg: A new graph-theoretic approach to collaborative filtering,” in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge discovery and data mining*. ACM Press, 1999, pp. 201–212.

[8] E. Alfonseca and S. Manandhar, “Extending a lexical ontology by a combination of distributional semantics signatures,” in *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, ser. EKAW ’02. Springer-Verlag, 2002, pp. 1–7.

[9] A. Almeida, B. Sotomayor, J. Abaitua, and D. López-de Ipiña, “folk2onto: Bridging the gap between social tags and ontologies,” in *International Workshop on Knowledge Reuse and Reengineering over the Semantic Web. hosted by the 5th European Semantic Web Conference (ESWC)*, 2008.

[10] M. Ames and M. Naaman, “Why we tag: motivations for annotation in mobile and online media,” in *CHI ’07: Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2007, pp. 971–980.

[11] S. S. Anand and B. Mobasher, “Introduction to intelligent techniques for web personalization,” *ACM Trans. Interet Technol.*, vol. 7, no. 4, p. 18, 2007.

[12] N. Aussenac-Gilles, B. Biebow, and S. Szulman, “Revisiting ontology design: A methodology based on corpus analysis,” in *EKAW ’00: Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*. London, UK: Springer-Verlag, 2000, pp. 172–188.

[13] L. Balby Marinho, K. Buza, and L. Schmidt-Thieme, “Folksonomy-based collabulary learning,” in *ISWC ’08: Proceedings of the 7th International Conference on The Semantic Web*. Springer-Verlag, 2008, pp. 261–276.

[14] S. Bao, G. Xue, X. Wu, Y. Yu, B. Fei, and Z. Su, “Optimizing web search using social annotations,” *Proceedings of the 16th international conference on World Wide Web*, pp. 501–510, 2007.

[15] P. Basile, D. Gendarmi, F. Lanubile, and G. Semeraro, “Recommending smart tags in a social bookmarking system,” in *Bridging the Gep between Semantic Web and Web 2.0 (SemNet 2007)*, 2007, pp. 22–29. [Online]. Available: http://www.kde.cs.uni-kassel.de/ws/eswc2007/proc/RecommendingSmartTags.pdf

[16] V. Batagelj and M. Zaveršnik, “Generalized cores,” *Arxiv preprint cs/0202039*, 2002.

[17] S. Berkovsky, L. Aroyo, D. Heckmann, G.-J. Houben, A. Kröner, T. Kuflik, and F. Ricci, "Providing context-aware personalization through cross-context reasoning of user modeling data," in *UbiDeUM'2007 - International Workshop on Ubiquitous and Decentralized User Modeling, at User Modeling 2007, 11th International Conference, UM 2007, Corfu, Greece, June 26, 2007, Proceedings*, S. Berkovsky, K. Cheverst, P. Dolog, D. Heckmann, T. Kuflik, P. Mylonas, J. Picault, and J. Vassileva, Eds., 2007. [Online]. Available: ./papers/UM-cross-context-v12.pdf

[18] B. Bi, L. Shang, and B. Kao, "Collaborative resource discovery in social tagging systems," in *CIKM '09: Proceeding of the 18th ACM conference on Information and knowledge management*. ACM, 2009, pp. 1919–1922.

[19] M. Bianchini, M. Gori, and F. Scarselli, "Inside pagerank," *ACM Trans. Interet Technol.*, vol. 5, no. 1, pp. 92–128, 2005.

[20] C. Biemann, "Ontology learning from text: A survey of methods," *LDV Forum*, vol. 20, no. 2, pp. 75–93, 2005.

[21] D. Billsus and M. J. Pazzani, "User modeling for adaptive news access," *User Modeling and User-Adapted Interaction*, vol. 10, no. 2-3, pp. 147–180, 2000.

[22] A. M. Bogers, "Recommender systems for social bookmarking," Ph.D. dissertation, Universiteit van Tilburg, 2009.

[23] G. Boone, "Concept features in re:agent, an intelligent email agent," in *Proceedings of the Second International Conference on Autonomous Agents*. ACM Press, 1998, pp. 141–148.

[24] S. Braun, A. Schmidt, and A. Walter, "Ontology maturing: a collaborative web 2.0 approach to ontology engineering," *Proceedings of the Workshop on Social and Collaborative Construction of Structured Knowledge (CKC) at the 16th International World Wide Web Conference (WWW 2007), year=2007,*.

[25] S. Braun, C. Kunzmann, and A. Schmidt, "People tagging & ontology maturing: Towards collaborative competence management," in *From CSCW to Web2.0: European Developments in Collaborative Design Selected Papers from COOP08*, ser. Computer Supported Cooperative Work, D. Randall and P. Salembier, Eds. Berlin/Heidelberg: Springer, 2010.

[26] S. Braun, C. Schora, and V. Zacharias, "Semantics to the Bookmarks: A Review of Social Semantic Bookmarking Systems," in *Proc. of the 5th I-SEMANTICS*, 2009, pp. 445–454.

[27] J. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” in *Uncertainty in Artificial Intelligence. Proceedings of the Fourteenth Conference*. Morgan Kaufman, 1998, pp. 43–53.

[28] J. S. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” in *Proceedings of the 14th Conference on Uncertainty in Artiffcial Intelligence*, 1998, pp. 43–52. [Online]. Available: http://citeseer.ist.psu.edu/breese98empirical.html

[29] S. Brin and L. Page, “The anatomy of a large-scale hypertextual web search engine,” *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107–117, 1998.

[30] J. Budzik, K. Hammond, and L. Birnbaum, “Information access in context,” *Knowledge based systems*, vol. 14, no. 1-2, pp. 37–53, 2001.

[31] P. Buono, M. F. Costabile, T. Guida, and A. Piccinno, “Integrating user data and collaborative filtering in a web recommendation system,” in *Lecture Notes in Computer Science: Hypermedia: Openness, Structural Awareness, and Adaptivity*. Springer-Link, 2002, pp. 192–196.

[32] R. Burke, “Hybrid recommender systems: Survey and experiments,” *User Modeling and User Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[33] R. Burke, B. Mobasher, and R. Bhaumik, “Limited knowledge shilling attacks in collaborative filtering systems,” in *Proceedings of the 3rd IJCAI Workshop in Intelligent Techniques for Personalization*, Edinburgh, Scotland, August 2005. [Online]. Available: http://maya.cs.depaul.edu/~mobasher/papers/sp-itwp05.pdf

[34] R. Burke, B. Mobasher, R. Zabicki, and R. Bhaumik, “Identifying attack models for secure recommendation,” in *Beyond Personalization: A Workshop on the Next Generation of Recommender Systems*, January 2005.

[35] R. Burke, “Knowledge-based recommender systems,” in *Encyclopedia of Library and Information Systems*. Marcel Dekker, 2000.

[36] ——, “Hybrid recommender systems: Survey and experiments,” *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002.

[37] ——, “Interactive critiquing for catalog navigation in e-commerce,” *Artif. Intell. Rev.*, vol. 18, no. 3-4, pp. 245–267, 2002.

[38] R. Burke, B. Mobasher, C. Williams, and R. Bhaumik, "Classification features for attack detection in collaborative recommender systems," in *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'06)*, Philadelphia, August 2006.

[39] R. Burke and M. Ramezani, *Matching Recommendation Technologies and Domains*. Springer, 2011.

[40] R. D. Burke, K. J. Hammond, and B. C. Young, "The findme approach to assisted browsing," *IEEE Expert: Intelligent Systems and Their Applications*, vol. 12, no. 4, pp. 32–40, 1997.

[41] D. Cameron, B. Aleman-meza, S. L. Decker, and I. B. Arpinar, "Semef: A taxonomy-based discovery of experts, expertise and collaboration networks," Tech. Rep., 2007.

[42] J. Canny, "Collaborative filtering with privacy via factor analysis," in *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 2002, pp. 238–245.

[43] L. C. C. Carvalheira and E. S. Gomi, "A method for semi-automatic creation of ontologies based on texts," in *Advances in Conceptual Modeling Foundations and Applications*. Springer, 2007, pp. 150–159.

[44] C. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher, "Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes," in *RE '08: Proceedings of the 2008 16th IEEE International Requirements Engineering Conference*. IEEE Computer Society, 2008, pp. 165–168.

[45] H.-C. Chen and A. L. P. Chen, "A music recommendation system based on music data grouping and user interests," in *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*. ACM, 2001, pp. 231–238.

[46] J. Chen and Q. Li, "Concept hierarchy construction by combining spectral clustering and subsumption estimation," in *Lecture Notes in Computer Science: Web Information Systems WISE 2006*. Springer-Verlag, 2006, pp. 199–209.

[47] E. H. Chi and T. Mytkowicz, "Understanding Navigability of Social Tagging Systems," *Proceedings of CHI*, vol. 7, 2007.

[48] P. Cimiano, A. Maedche, S. Staab, and J. Voelker, *Ontology Learning*, 2nd ed., ser. International Handbooks on Information Systems, S. Staab and R. Studer, Eds. Springer, 2009.

[49] P. Cimiano, A. Pivk, L. S. Thieme, and S. Staab, "Learning taxonomic relations from heterogeneous sources of evidence," in *Proceedings of the ECAI 2004 Ontology Learning and Population Workshop*, 2004.

[50] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, "Combining content-based and collaborative filters in an online newspaper," in *Proceedings of the ACM SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation*, Berkeley, California, August 1999.

[51] M. Clements, "Personalization of social media," in *Proceedings of the BCS IRSG Symposium: Future Directions in Information Access*, 2007.

[52] W. W. Cohen, P. Ravikumar, and S. E. Fienberg, "A Comparison of String Distance Metrics for Name-Matching Tasks," in *Proc. of IJCAI'03 Workshop on Information Integration on the Web*, 2003, pp. 73–78.

[53] A. M. Collins and M. R. Quillian, "Retrieval time from semantic memory," *Journal of Verbal Learning and Verbal Behavior*, vol. 8, pp. 240–248, 1969.

[54] R. Cooley, B. Mobasher, and J. Srivastava, "Web mining: Information and pattern discovery on the world wide web," in *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, Newport Beach, CA, November 1997, pp. 558–567.

[55] M. Cristani and R. Cuel, "A survey on ontology creation methodologies," *International Journal of Semantic Web Information Systems*, vol. 1, no. 2, pp. 49–69, 2005.

[56] A. S. Das, M. Datar, A. Garg, and S. Rajaram, "Google news personalization: scalable online collaborative filtering," in *WWW '07: Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 271–280.

[57] A. K. Dey, "Understanding and using context," *Personal and Ubiquitous Computing*, vol. 5, no. 1, pp. 4–7, 2001.

[58] J. Diederich and T. Iofciu, "Finding Communities of Practice from User Profiles Based On Folksonomies," *Proceedings of the 1st International Workshop on Building Technology Enhanced Learning solutions for Communities of Practice (TEL-CoPs 06),*

co-located with the First European Conference on Technology-Enhanced Learning, Crete, Greece, 2006.

[59] F. Echarte, J. J. Astrain, A. Córdoba, and J. E. Villadangos, "Ontology of folksonomy: A new modelling method," in *Proceedings of Semantic Authoring, Annotation and Knowledge Markup Workshop(SAAKM) co-located with the Fourth International Conference on Knowledge Capture K-Cap*, 2007.

[60] F. Eisterlehner, A. Hotho, and R. Jaeschke, Eds., *ECML PKDD Discovery Challenge 2009 (DC09)*, ser. CEUR-WS.org, vol. 497, Sep. 2009. [Online]. Available: http://ceur-ws.org/Vol-497

[61] A. Felfernig and R. Burke, "Constraint-based recommender systems: technologies and research issues," in *ICEC '08: Proceedings of the 10th international conference on Electronic commerce*. ACM, 2008, pp. 1–10.

[62] A. Felfernig, "Koba4ms: Selling complex products and services using knowledge-based recommender technologies," in *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*. IEEE Computer Society, 2005, pp. 92–100.

[63] A. Felfernig and A. Kiener, "Knowledge-based interactive selling of financial services with fsadvisor," in *Proceedings of the National Conference on Artificial Intelligence*, 2005, pp. 1475–1482.

[64] P. L. Fisher-Ogden, "Def-cat: Dblp expert finder utilizing categories and topics," Master's thesis, University of California, San Diego, 2001.

[65] M. Fleischman and E. Hovy, "Fine grained classification of named entities," in *Proceedings of the 19th International Conference on Computational Linguistics*, 2002, pp. 1–7.

[66] N. Garg and I. Weber, "Personalized, interactive tag recommendation for flickr," in *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*. New York, NY, USA: ACM, 2008, pp. 67–74.

[67] J. Gemmell, A. Shepitsen, B. Mobasher, and R. Burke, "Personalization in Folksonomies Based on Tag Clustering," *Intelligent Techniques for Web Personalization & Recommender Systems*, 2008.

[68] ——, “Personalizing navigation in folksonomies using hierarchical tag clustering,” in *Proceedings of the 10th international conference on Data Warehousing and Knowledge Discovery*. Springer, 2008.

[69] J. Gemmell, M. Ramezani, T. Schimoler, L. Christiansen, and B. Mobasher, “The impact of ambiguity and redundancy on tag recommendation in folksonomies,” in *RecSys ’09: Proceedings of the third ACM conference on Recommender systems*. ACM, 2009, pp. 45–52.

[70] J. Gemmell, T. Schimoler, M. Ramezani, and B. Mobasher, “Adapting k-nearest neighbor for tag recommendation in folksonomies,” *Intelligent Techniques for Web Personalization & Recommender Systems*, 2009.

[71] J. Gemmell, A. Shepitsen, B. Mobasher, and R. Burke, “Personalizing navigation in folksonomies using hierarchical tag clustering,” in *Proceedings of the 10th International Conference on Data Warehousing and Knowledge Discovery*. Springer, 2008.

[72] ——, “Personalization in Folksonomies Based on Tag Clustering,” in *Intelligent Techniques for Web Personalization & Recommender Systems*, 2008.

[73] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, “Using collaborative filtering to weave an information tapestry,” *Communications of the ACM*, vol. 35, pp. 61–70, 1992.

[74] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, “Eigentaste: A constant time collaborative filtering algorithm,” *Inf. Retr.*, vol. 4, no. 2, pp. 133–151, July 2001.

[75] S. Golder and B. A. Huberman, “The Structure of Collaborative Tagging Systems,” *Arxiv preprint cs.DL/0508082*, 2005.

[76] S. A. Golder and B. A. Huberman, “Usage patterns of collaborative tagging systems,” *Journal of Information Science*, vol. 32, no. 2, pp. 198–208, 2006.

[77] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. M. Sarwar, J. L. Herlocker, and J. Riedl, “Combining collaborative filtering with personal agents for better recommendations,” in *AAAI:Conference on Artificial Intelligence*, 1999, pp. 439–446. [Online]. Available: citeseer.ist.psu.edu/good99combining.html

[78] T. Gruber, “Ontology of folksonomy: A mash-up of apples and oranges,” in *First on-Line conference on Metadata and Semantics Research (MTSR’05)*, 2005.

[79] T. R. Gruber, “A translation approach to portable ontology specifications,” *Knowl. Acquis.*, vol. 5, pp. 199–220, June 1993.

[80] N. Guarino, R. Poli, K. A. Publishers, I. P. Substantial, and T. R. Gruber, “Toward principles for the design of ontologies used for knowledge sharing,” in *In Formal Ontology in Conceptual Analysis and Knowledge Representation, Kluwer Academic Publishers, in press.*, 1993.

[81] S. H. Ha, “Digital content recommender on the internet,” *IEEE Intelligent Systems*, vol. 21, no. 2, pp. 70–77, 2006.

[82] T. Hammond, T. Hannay, B. Lund, and J. Scott, “Social bookmarking tools.”

[83] J. Han, , J. Han, and Y. Fu, “Dynamic generation and refinement of concept hierarchies for knowledge discovery in databases,” in *Proceedings of AAAI’94 Workshop on Knowledge Discovery in Databases (KDD’94*, 1994, pp. 157–168.

[84] C. Hayes and P. Cunningham, “Smart radio: Building music radio on the fly,” in *Proceedings of Expert Systems 2000, Cambridge, UK*, 2000.

[85] C. Hayes, P. Avesani, and S. Veeramachaneni, “An analysis of the use of tags in a blog recommender system,” in *IJCAI-07, the International Joint Conference on Artificial Intelligence*, M. M. Veloso, Ed., 2007, pp. 2772–2777.

[86] M. A. Hearst, “Automatic acquisition of hyponyms from large text corpora,” in *Proceedings of the 14th conference on Computational linguistics - Volume 2*, ser. COLING ’92. Association for Computational Linguistics, 1992, pp. 539–545.

[87] M. Hepp, “Possible ontologies: How reality constrains the development of relevant ontologies,” *IEEE Internet Computing*, vol. 11, pp. 90–96, 2007.

[88] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl, “An algorithmic framework for performing collaborative filtering,” in *Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval (SIGIR’99)*, Berkeley, CA, August 1999.

[89] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, “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*. New York, NY, USA: ACM Press, 1999, pp. 230–237.

[90] J. L. Herlocker, J. A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *CSCW '00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*. ACM Press, 2000, pp. 241–250.

[91] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 5–53, 2004.

[92] P. Heymann, G. Koutrika, and H. Garcia-Molina, "Fighting Spam on Social Web Sites: A Survey of Approaches and Future Challenges," *IEEE Internet Computing*, vol. 11, no. 6, pp. 36–45, 2007.

[93] P. Heymann and H. Garcia-Molina, "Collaborative creation of communal hierarchical taxonomies in social tagging systems," Stanford University, Tech. Rep. 2006-10, April 2006. [Online]. Available: http://heymann.stanford.edu/taghierarchy.html

[94] P. Heymann, D. Ramage, and H. Garcia-Molina, "Social tag prediction," in *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, 2008, pp. 531–538.

[95] Y. Hijikata, K. Iwahama, and S. Nishida, "Content-based music filtering system with editable user profile," in *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*. ACM, 2006, pp. 1050–1057.

[96] A. Hippisley, D. Cheng, and K. Ahmad, "The head-modifier principle and multilingual term extraction," *Natural Language Engineering*, vol. 11, no. 2, pp. 129–157, 2005.

[97] T. Hofmann, "Latent semantic models for collaborative filtering," *ACM Transactions on Information Systems*, vol. 22, no. 1, pp. 89–115, 2004.

[98] R. Holmes, R. J. Walker, and G. C. Murphy, "Approximate structural context matching: An approach to recommend relevant examples," *IEEE Transactions on Software Engineering*, vol. 32, no. 12, pp. 952–970, 2006.

[99] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme, "Emergent Semantics in BibSonomy," *Proceedings of Workshop on Applications of Semantic Technologies, Informatik*, 2006.

[100] A. Hotho, R. Jaschke, C. Schmitz, and G. Stumme, "Information retrieval in folksonomies: Search and ranking," *The Semantic Web: Research and Applications*, vol. 4011, pp. 411–426, 2006.

[101] A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme, "Folkrank: A ranking algorithm for folksonomies," in *Proc. FGIR 2006*, 2006. [Online]. Available: http://www.kde.cs.uni-kassel.de/stumme/papers/2006/hotho2006folkrank.pdf

[102] ——, "Information retrieval in folksonomies: Search and ranking," in *The Semantic Web: Research and Applications*, 2006, pp. 411–426.

[103] S. hyung Hwang, H.-G. Kim, M.-K. Kim, S.-H. Choi, and H. S. Yang, "A data-driven approach to constructing an ontological concept hierarchy based on the formal concept analysis," in *Computational Science and Its Applications - ICCSA 2006*. Springer, 2006, pp. 937–946.

[104] J. Illig, A. Hotho, R. Jaeschke, and G. Stumme, "A comparison of content-based tag recommendations in folksonomy systems," in *Postproceedings of the International Conference on Knowledge Processing in Practice (KPP 2007)*. Springer, 2009.

[105] R. Jäschke, L. B. Marinho, A. Hotho, L. Schmidt-Thieme, and G. Stumme, "Tag recommendations in folksonomies," in *Knowledge Discovery in Databases: PKDD 2007*, ser. Lecture Notes in Computer Science, vol. 4702. Springer, 2007, pp. 506–514.

[106] S. T. Jeff Z. Pan and E. Thomas, "Reducing ambiguity in tagging systems with folksonomy search expansion," in *6th European Semantic Web Conference 2009*, 2009.

[107] A. John, "Collaborative tagging and expertise in the enterprise," in *Proceedings of Collaborative Web Tagging Workshop held in Conjunction with WWW 2006*, 2006.

[108] M. E. I. Kipp and G. D. Campbell, "Patterns and inconsistencies in collaborative tagging systems : An examination of tagging practices," *Proceedings of the American Society for Information Science and Technology*, November 2006. [Online]. Available: http://eprints.rclis.org/archive/00008315/

[109] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.

[110] R. Kosala and H. Blockeel, "Web mining research: a survey," *SIGKDD Explor. Newsl.*, vol. 2, no. 1, pp. 1–15, 2000.

[111] G. Koutrika, F. A. Effendi, Z. Gyöngyi, P. Heymann, and H. Garcia-Molina, "Combating spam in tagging systems," *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, pp. 57–64, 2007.

[112] A. I. Kovacs and H. Ueno, "Recommending in context: A spreading activation model that is independent of the type of recommender system and its contents," in *Proceedings of Workshop on Web Personalisation, Recommender Systems and Intelligent User Interfaces In conjunction with AH 2006:International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, G. Uchyigit, Ed., Dublin, Ireland, 2006.

[113] B. Krause, C. Schmitz, A. Hotho, and G. Stumme, "The anti-social tagger - detecting spam in social bookmarking systems," in *Proceedings of the Fourth International Workshop on Adversarial Information Retrieval on the Web*, 2008.

[114] M. Krótzsch, D. Vrandecic, M. Vólkel, H. Haller, and R. Studer, "Semantic Wikipedia," *Journal of Web Semantics*, vol. 5, pp. 251–261, 2007.

[115] B. Krulwich, "Lifestyle finder: Intelligent user profiling using large-scale demographic data," *Artificial Intelligence Magazine*, vol. 18, no. 2, 1997.

[116] M. Kurucz, A. A. Benczúr, T. Kiss, I. Nagy, A. Szabó, and B. Torma, "Kdd cup 2007 task 1 winner report," *SIGKDD Explor. Newsl.*, vol. 9, no. 2, pp. 53–56, 2007.

[117] S. Lam and J. Riedl, "Shilling recommender systems for fun and profit," in *Proceedings of the 13th International WWW Conference*, New York, May 2004.

[118] K. Lang, "Newsweeder: Learning to filter news," in *Proceedings of the 12th International Conference on Machine Learning*, 1995, pp. 331–339.

[119] D. H. Lee and P. Brusilovsky, "Fighting information overflow with personalized comprehensive information access: A proactive job recommender," in *ICAS '07: Proceedings of the Third International Conference on Autonomic and Autonomous Systems*. IEEE Computer Society, 2007, p. 21.

[120] J. Leskovec, "Dynamics of large networks," Ph.D. dissertation, Stanford University, 2008.

[121] J. Li and O. R. Zaiane, "Combining usage, content, and structure data to improve web site recommendation," *Lecture Notes in Computer Science : E-Commerce and Web Technologies*, pp. 305–315, 2004.

[122] M. Lipczak, "Tag recommendation for folksonomies oriented towards individual users," in *Proceedings of the ECML/PKDD 2008 Discovery Challenge Workshop, part of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2008.

[123] M. Lipczak, Y. Hu, Y. Kollet, and E. Milios, “Tag sources for recommendation in collaborative tagging systems,” in *ECML PKDD Discovery Challenge 2009 (DC09)*, F. Eisterlehner, A. Hotho, and R. Jaeschke, Eds., vol. 497. Bled, Slovenia: CEUR Workshop Proceedings, 2009, pp. 157–172. [Online]. Available: http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-497/

[124] G. Macgregor and E. McCulloch, “Collaborative tagging as a knowledge organisation and resource discovery tool,” *Library Review*, vol. 55, no. 5, pp. 291–300, 2006.

[125] A. Maedche and S. Staab, “Mining ontologies from text,” in *Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*, ser. EKAW ’00. Springer-Verlag, 2000, pp. 189–202. [Online]. Available: http://portal.acm.org/citation.cfm?id=645361.757522

[126] ——, “Ontology learning for the semantic web,” *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 72–79, 2001.

[127] T. Mahmood, F. Ricci, A. Venturini, and W. Höpken, “Adaptive recommender systems for travel planning.” in *Information and Communication Technologies in Tourism 2008*, P. O’Connor, W. Höpken, and U. Gretzel, Eds. Springer, 2008, pp. 1–11.

[128] V. Maidel, P. Shoval, B. Shapira, and M. Taieb-Maimon, “Evaluation of an ontology-content based filtering method for a personalized newspaper,” in *RecSys ’08: Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 2008, pp. 91–98.

[129] O. Malik, “del.icio.us popular is spammed,” http://gigaom.com/2006/06/05/delicious-popular-being-spammed/, 2006.

[130] J. Malinowski, T. Keim, O. Wendt, and T. Weitzel, “Matching people and jobs: A bilateral recommendation approach,” in *HICSS ’06: Proceedings of the 39th Annual Hawaii International Conference on System Sciences*. IEEE Computer Society, 2006, p. 137c.

[131] B. Markines, L. Stoilova, and F. Menczer, “Bookmark hierarchies and collaborative recommendation,” in *Proceedings of the Twenty-First AAAI Conference on Artificial Intelligence*. AAAI Press, 2006.

[132] B. Markines, C. Cattuto, F. Menczer, D. Benz, A. Hotho, and G. Stumme, “Evaluating similarity measures for emergent semantics of social tagging,” in *WWW ’09: Proceedings of the 18th international conference on World wide web*. ACM, 2009, pp. 641–650.

[133] C. Marlow, M. Naaman, D. Boyd, and M. Davis, "HT06, tagging paper, taxonomy, Flickr, academic article, to read," *Proceedings of the seventeenth conference on Hypertext and hypermedia*, pp. 31–40, 2006.

[134] A. Mathes, "Folksonomies-Cooperative Classification and Communication Through Shared Metadata," *Computer Mediated Communication, (Doctoral Seminar), Graduate School of Library and Information Science, University of Illinois Urbana-Champaign, December*, 2004.

[135] D. McSherry, "Explaining the pros and cons of conclusions in cbr," in *Proceedings of the European Conference on Case-Based Reasoning (ECCBR-04)*, P. A. G. Calero and P. Funk, Eds. Springer, 2004, pp. 317–330, madrid, Spain.

[136] ——, "Explanation in recommender systems," *Artif. Intell. Rev.*, vol. 24, no. 2, pp. 179–197, 2005.

[137] D. McSherry and D. W. Aha, "Mixed-initiative relaxation of constraints in critiquing dialogues," in *ICCBR '07: Proceedings of the 7th international conference on Case-Based Reasoning*, vol. 4626, 2007, pp. 107–121.

[138] P. Mika, "Ontologies are us: A unified model of social networks and semantics," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 5, no. 1, pp. 5–15, 2007.

[139] ——, "Ontologies are us: A unified model of social networks and semantics," in *In International Semantic Web Conference*, 2005, pp. 522–536.

[140] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller, "Wordnet: An on-line lexical database," *International Journal of Lexicography*, vol. 3, pp. 235–244, 1990.

[141] D. Mladenic and M. Grobelnik, "Feature selection for unbalanced class distribution and naive bayes," in *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 258–267.

[142] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, "Effective attack models for shilling item-based collaborative filtering systems," in *Proceedings of the 2005 WebKDD Workshop, held in conjuction with ACM SIGKDD'2005*, Chicago, Illinois, August 2005.

[143] B. Mobasher, R. Burke, R. Bhaumik, and J. J. Sandvig, “Attacks and remedies in collaborative recommendation,” *IEEE Intelligent Systems*, vol. 22, no. 3, pp. 56–63, 2007.

[144] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, “Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness,” *ACM Transaction on Internet Technology*, vol. 7, no. 4, p. 23, 2007.

[145] B. Mobasher, R. Cooley, and J. Srivastava, “Automatic personalization based on web usage mining,” *Communications of the ACM*, vol. 43, no. 8, pp. 142–151, 2000.

[146] M. Montaner, B. López, and J. L. D. L. Rosa, “A taxonomy of recommender agents on the internet,” *Artif. Intell. Rev.*, vol. 19, no. 4, pp. 285–330, 2003.

[147] R. J. Mooney and L. Roy, “Content-based book recommending using learning for text categorization,” in *DL ’00: Proceedings of the fifth ACM conference on Digital libraries*. ACM Press, 2000, pp. 195–204.

[148] E. Moreau, F. Yvon, and O. Cappé, “Robust similarity measures for named entities matching,” in *COLING ’08: Proceedings of the 22nd International Conference on Computational Linguistics*. Association for Computational Linguistics, 2008, pp. 593–600.

[149] P. J. Morrison, “Tagging and searching: Search retrieval effectiveness of folksonomies on the world wide web,” *Information Processing and Management*, vol. 44, no. 4, pp. 1562 – 1579, 2008.

[150] R. Y. Nakamoto, S. Nakajima, J. Miyazaki, S. Uemura, and H. Kato, “Investigation of the effectiveness of tag-based contextual collaborative filtering in website recommendation,” in *Advances in Communication Systems and Electrical Engineering*. Springerlink, 2008, pp. 309–318.

[151] R. Y. Nakamoto, S. Nakajima, J. Miyazaki, S. Uemura, H. Kato, and Y. Inagaki, “Reasonable tag-based collaborative filtering for social tagging systems,” in *WICOW ’08: Proceeding of the 2nd ACM workshop on Information credibility on the web*. Napa Valley, California, USA: ACM, 2008, pp. 11–18.

[152] M. Nauman, S. Khan, M. Amin, and F. Hussain, “Resolving Lexical Ambiguities in Folksonomy Based Search Systems through Common Sense and Personalization,” in *Proceedings of the Workshop on Semantic Search at the 5th European Semantic Web Conference, Tenerife, Spain*, 2008, pp. 2–13.

[153] S. Niwa, T. Doi, and S. Honiden, “Web Page Recommender System based on Folksonomy Mining for ITNG06 Submissions,” *Proceedings of the Third International Conference on Information Technology: New Generations*, pp. 388–393, 2006.

[154] N. F. Noy and D. L. McGuinness, “Ontology development 101: A guide to creating your first ontology,” Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, Tech. Rep., 2001.

[155] M. O’Mahony, N. Hurley, N. Kushmerick, and G. Silvestre, “Collaborative recommendation: A robustness analysis,” *ACM Transactions on Internet Technology*, vol. 4, no. 4, pp. 344–377, 2004.

[156] J. Z. Pan, S. Taylor, and E. Thomas, “Reducing ambiguity in tagging systems with folksonomy search expansion,” in *Proceedings of the 6th European Semantic Web Conference on The Semantic Web: Research and Applications*, ser. ESWC 2009 Heraklion. Springer-Verlag, 2009, pp. 669–683.

[157] A. Paterek, “Improving regularized singular value decomposition for collaborative filtering,” in *Proc. of the of the KDD Cup and Workshop 2007 (KDD 2007)*, August 2007.

[158] B. M. Paul Buitelaar, Philipp Cimiano, *Ontology learning from text: methods, evaluation and applications*. IOS Press, 2005.

[159] M. Pazzani and D. Billsus, “Learning and revising user profiles: The identification ofinteresting web sites,” *Machine Learning: Special issue on multistrategy learning*, vol. 27, no. 3, pp. 313–331, 1997.

[160] M. Pazzani, J. Muramatsu, and D. Billsus, “Syskill & webert: Identifying interesting web sites,” in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*. AAAI Press, 1996, pp. 54–61.

[161] M. J. Pazzani, “A framework for collaborative, content-based and demographic filtering,” *Artif. Intell. Rev.*, vol. 13, no. 5-6, pp. 393–408, 1999.

[162] A. Plangprasopchok and K. Lerman, “Exploiting Social Annotation for Automatic Resource Discovery,” *eprint arXiv: 0704.1675*, 2007.

[163] M. H. Pryor, “The effects of singular value decomposition on collaborative filtering,” Hanover, NH, USA, Tech. Rep., 1998.

[164] B. R, M. B, W. C, and B. R, “Detecting profile injection attacks in collaborative recommender systems.” *In: Proceedings of the IEEE joint conference on e-commerce technology and enterprise computing, e-commerce and e-services (CEC/EEE 2006), Palo Alto, CA, June 2006k*, 2006.

[165] C. Rack, S. Arbanowski, and S. Steglich, *A Generic Multipurpose recommender System for Contextual Recommendations*. IEEE Computer Society, 2007, pp. 445–450.

[166] M. Ramezani, “Improving graph-based approaches for personalized tag recommendation,” *Journal of Emerging Technologies in Web Intelligence (JETWI)*, vol. To appear, 2011.

[167] M. Ramezani, L. Bergman, R. Thompson, R. Burke, and B. Mobasher, “Selecting and Applying Recommendation Technology,” in *International Workshop on Recommendation and Collaboration in Conjunction with 2008 International ACM Conference on Intelligent User Interfaces (IUI 2008)*, 2008.

[168] M. Ramezani, H. FriedrichWitschel, S. Braun, and V. Zacharias, “Using machine learning to support continuous ontology development,” in *EKAW 2010:17th International Conference on Knowledge Engineering and Knowledge Management*, 2010.

[169] M. Ramezani, J. Gemmell, T. Schimoler, and B. Mobasher, “Improving link analysis for tag recommendation in folksonomies,” in *International Workshop on Recommender Systems and the Social Web, Recsys 10*, 2010.

[170] M. Ramezani, J. J. Sandvig, R. Bhaumik, R. Burke, and B. Mobasher, “Exploring the impact of profile injection attacks in social tagging systems,” in *Proceedings of WebKDD*, 2008.

[171] M. Ramezani, J. J. Sandvig, T. Schimoler, J. Gemmell, B. Mobasher, and R. Burke, “Evaluating the impact of attacks in collaborative tagging environments,” in *IEEE International Conference on Social Computing, SocialCom09*. IEEE Computer Society, 2009, pp. 136–143.

[172] M. Ramezani, A. Shepitsen, R. Bhaumik, R. Burke, and B. Mobasher, “Using semantic cues for contextual recomemdnation,” Proceedings of the 2007 DePaul CTI Research symposium, 2007.

[173] M. Ramezani and H. F. Witschel, “An intelligent system for semi-automatic evolution of ontologies,” in *Proceedings of 5th IEEE International Conference on Intelligent Systems IS10*, 2010.

[174] S. Rendle and L. Schmidt-Thieme, “Pairwise interaction tensor factorization for personalized tag recommendation,” in *WSDM ’10: Proceedings of the third ACM international conference on Web search and data mining*. New York, NY, USA: ACM, 2010, pp. 81–90.

[175] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “Grouplens: an open architecture for collaborative filtering of netnews,” in *CSCW ’94: Proceedings of the 1994 ACM conference on Computer supported cooperative work*. Chapel Hill, NC: ACM Press, 1994, pp. 175–186.

[176] P. Resnick and H. R. Varian, “Recommender systems,” *Commun. ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[177] F. Ricci, “Travel recommender systems,” in *IEEE Intelligent Systems*, 2002, pp. 55–57.

[178] A. Riska, V. Diev, and E. Smirni, “Efficient fitting of long-tailed data sets into hyperexponential distributions,” in *IEEE Globecom Conference, Internet Performance Symposium, Taipei, Taiwan, November 2002. IEEE Catalog Number: 02CH3798C*, 2002.

[179] S. Rosset, C. Perlich, and Y. Liu, “Making the most of your data: Kdd cup 2007 ”how many ratings” winner’s report,” *SIGKDD Explor. Newsl.*, vol. 9, no. 2, pp. 66–69, 2007.

[180] T. R. Roth-Berghofer, “Explanations and case-based reasoning: Foundational issues,” in *AAdvances in Case-Based Reasoning*. Springer Verlag, 2004, pp. 389–403.

[181] M. Salam, J. Reilly, L. McGinty, and B. Smyth, “Knowledge discovery from user preferences in conversational recommendation,” in *Knowledge Discovery in Databases: PKDD 2005*. Springer Berlin / Heidelberg, 2005, pp. 228–239.

[182] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information Processing and Management: an International Journal*, vol. 24, no. 5, pp. 513–523, 1988.

[183] G. Salton, A. Wong, and C. Yang, “A vector space model for automatic indexing,” *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.

[184] M. Sanderson and B. Croft, “Deriving concept hierarchies from text,” in *Proceedings of the 22nd ACM SIGIR Conference*, Berkeley, California, 1999, pp. 206–213.

[185] J. J. Sandvig, B. Mobasher, and R. Burke., "Robustness of collaborative recommendation based on association rule mining." *Proceedings of the 2007 ACM Conference on Recommender Systems, Minneapolis, October*, 2007.

[186] J. Sandvig, R. Bhaumik, M. Ramezani, R. Burke, and B. Mobasher, "A Framework for the Analysis of Attacks Against Social Tagging Systems," in *The 6th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems, held in conjunction with The 23nd National Conference on Artificial Intelligence - AAAI 2008*, Chicago, Illinois, USA, 2008.

[187] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender systems-a case study," in *Proceedings of ACM WebKDD Workshop*, 2000.

[188] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *WWW '01: Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 285–295.

[189] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Incremental singular value decomposition algorithms for highly scalable recommender systems," in *Fifth International Conference on Computer and Information Science*, 2002, pp. 27–28.

[190] J. B. Schafer, J. A. Konstan, and J. Riedl, "E-commerce recommendation applications," *Data Mining and Knowledge Discovery*, vol. 5, no. 1-2, pp. 115–153, 2001.

[191] C. Schmitz, A. Hotho, R. Jaschke, and G. Stumme, "Mining association rules in folksonomies," in *Proc. IFCS 2006 Conference*. Springer, 2006, pp. 261–270.

[192] U. Shardanand and P. Maes, "Social information filtering: algorithms for automating "word of mouth"," in *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM Press/Addison-Wesley Publishing Co., 1995, pp. 210–217.

[193] A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke, "Personalized recommendation in social tagging systems using hierarchical clustering," in *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*. New York, NY, USA: ACM, 2008, pp. 259–266.

[194] B. Sigurbjörnsson and R. van Zwol, "Flickr tag recommendation based on collective knowledge," pp. 327–336, 2008.

[195] K. Siorpaes and D. Bachlechner, “Harvesting wiki consensus - using wikipedia entries as ontology elements,” in *IEEE Internet Computing*, 2006, pp. 54–65.

[196] B. Smyth, E. Balfe, J. Freyne, P. Briggs, M. Coyle, and O. Boydell, “Exploiting query repetition and regularity in an adaptive community-based web search engine,” *User Modeling and User-Adapted Interaction*, vol. 14, no. 5, pp. 383–423, 2005.

[197] Y. Song, L. Zhang, and C. L. Giles, “A sparse gaussian processes classification framework for fast tag suggestions,” in *CIKM ’08: Proceeding of the 17th ACM conference on Information and knowledge management*. New York, NY, USA: ACM, 2008, pp. 93–102.

[198] Y. Song, Z. Zhuang, H. Li, Q. Zhao, J. Li, W.-C. Lee, and C. L. Giles, “Real-time automatic tag recommendation,” in *SIGIR ’08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. Singapore: ACM, 2008, pp. 515–522.

[199] J. F. Sowa, “Building, Sharing, and Merging Ontologies,” http://www.jfsowa.com/ontology/ontoshar.htm, 2009.

[200] L. Stojanovic, A. Maedche, B. Motik, and N. Stojanovic, “User-driven ontology evolution management,” in *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, ser. EKAW ’02. Springer-Verlag, 2002, pp. 285–300.

[201] Y. Sure, M. Erdmann, J. Angele, S. Staab, R. Studer, and D. Wenke, “OntoEdit: Collaborative Ontology Development for the Semantic Web,” in *First International Semantic Web Conference (ISWC 2002*, vol. 2342. Springer, 2002, pp. 221–235.

[202] G. Takács, I. Pilászy, B. Németh, and D. Tikk, “On the gravity recommendation system,” in *Proc. of the of the KDD Cup and Workshop 2007 (KDD 2007)*, August 2007, pp. 22–30.

[203] G. Takács, I. Pilászy, B. Németh, and D. Tikk, “Matrix factorization and neighbor based algorithms for the netflix prize problem,” in *RecSys ’08: Proceedings of the 2008 ACM conference on Recommender systems*. ACM, 2008, pp. 267–274.

[204] J. Tang, H. fung Leung, Q. Luo, D. Chen, and J. Gong, “Towards ontology learning from folksonomies,” in *IJCAI’09: Proceedings of the 21st international jont conference on Artifical intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009, pp. 2089–2094. [Online]. Available: http://ijcai.org/papers09/Papers/IJCAI09-344.pdf

[205] A. Tartakovski, M. Schaaf, and R. Bergmann, “Retrieval and configuration of life insurance policies,” in *Lecture Notes in Computer Science, Case-Based Reasoning Research and Development*. Springer Berlin / Heidelberg, 2005, pp. 552–565.

[206] K. H. L. Tso-Sutter, L. B. Marinho, and L. Schmidt-Thieme, “Tag-aware recommender systems by fusion of collaborative filtering algorithms,” in *SAC ’08: Proceedings of the 2008 ACM symposium on Applied computing*. ACM New York, NY, USA, 2008, pp. 1995–1999.

[207] L. Ungar, D. Foster, E. Andre, S. Wars, F. S. Wars, D. S. Wars, and J. H. Whispers, “Clustering methods for collaborative filtering,” in *Workshop on Recommender Systems at the 15th National Conference on Artificial Intelligence*. AAAI Press, 1998.

[208] F. user forum, “Exclude certain users from all searches,” http://www.flickr.com/groups/flickrideas/discuss/72157600044251686/, 2006.

[209] D. Vallet, I. Cantador, and J. M. Jose, “Personalizing web search with folksonomy-based user and document profiles,” in *ECIR 2010: Proceedings of the 32nd European Conference on Information Retrieval*, 2010, pp. 420–431.

[210] C. Van Damme, M. Hepp, and K. Siorpaes, “Folksontology: An integrated approach for turning folksonomies into ontologies,” in *Proceedings of the ESWC Workshop Bridging the Gap between Semantic Web and Web 2.0 (SemNet 2007)*, 2007, pp. 57–70.

[211] C. Van Rijsbergen, *Information Retrieval*. Butterworth-Heinemann Newton, MA, USA, 1979.

[212] M. van Setten, S. Pokraev, and J. Koolwaaij, “Context-aware recommendations in the mobile tourist application compass,” in *Adaptive Hypermedia 2004*, W. Nejdl and P. De Bra, Eds. Springer Verlag, 2004, pp. 235–244. [Online]. Available: ./papers/settenah2004.pdf

[213] P. Viappiani, P. Pu, and B. Faltings, “Conversational recommenders with adaptive suggestions,” in *RecSys ’07: Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 2007, pp. 89–96.

[214] J. Vig, S. Sen, and J. Riedl, “Tagsplanations: explaining recommendations using tags,” in *IUI ’09: Proceedings of the 13th international conference on Intelligent user interfaces*. New York, NY, USA: ACM, 2009, pp. 47–56.

[215] V. Wal, “Folksonomy coinage and definition,” http://vanderwal.net/folksonomy.html, 2004.

[216] Y. Wang, J. Vlker, and P. Haase, “Towards semi-automatic ontology building supported by large-scale knowledge acquisition,” in *AAAI Fall Symposium On Semantic Web for Collaborative Knowledge Acquisition*, vol. FS-06-06, AAAI. Arlington, VA, USA: AAAI Press, Oktober 2006, pp. 70–77.

[217] R. Wetzker, W. Umbrath, and A. Said, “A hybrid approach to item recommendation in folksonomies,” in *ESAIR ’09: Proceedings of the WSDM ’09 Workshop on Exploiting Semantic Annotations in Information Retrieval.* ACM, 2009, pp. 25–29.

[218] D. Widdows, “Unsupervised methods for developing taxonomies by combining syntactic and statistical information,” in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL ’03. Association for Computational Linguistics, 2003, pp. 197–204.

[219] W. E. Winkler, “The state of record linkage and current research problems,” Statistical Research Division, U.S. Census Bureau, Tech. Rep., 1999.

[220] H. F. Witschel, “Using decision trees and text mining techniques for extending taxonomies,” in *Proceedings of the Workshop on Learning and Extending Lexical Ontologies by using Machine Learning (OntoML)*, Bonn, Germany, 2005.

[221] H. Wu, M. Zubair, and K. Maly, “Harvesting social knowledge from folksonomies,” *Proceedings of the seventeenth conference on Hypertext and hypermedia*, pp. 111–114, 2006.

[222] X. Wu, L. Zhang, and Y. Yu, “Exploring social annotations for the semantic web,” in *WWW ’06: Proceedings of the 15th international conference on World Wide Web*. New York, NY, USA: ACM, 2006, pp. 417–426.

[223] X.-h. Xu, J.-l. Huang, J. Wan, and C.-f. Jiang, “A method for measuring semantic similarity of concepts in the same ontology,” in *IMSCCS ’08: Proceedings of the 2008 International Multi-symposiums on Computer and Computational Sciences.* IEEE Computer Society, 2008, pp. 207–213.

[224] Z. Xu, Y. Fu, J. Mao, and D. Su, “Towards the semantic web: Collaborative tag suggestions,” in *Collaborative Web Tagging Workshop in conjunction with the 15th WWW Conference*, Edinburgh, Scotland, May 2006.

[225] D. Yang and D. M. W. Powers, "Measuring semantic similarity in the taxonomy of wordnet," in *ACSC '05: Proceedings of the Twenty-eighth Australasian conference on Computer Science*. Australian Computer Society, Inc., 2005, pp. 315–322.

[226] C. Yeung, N. Gibbins, and N. Shadbolt, "Tag meaning disambiguation through analysis of tripartite structure of folksonomies," in *Proceedings of the 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops*. IEEE Computer Society Washington, DC, USA, 2007, pp. 3–6.

[227] ——, "Understanding the semantics of ambiguous tags in folksonomies," in *Proceedings of the International Workshop on Emergent Semantics and Ontology Evolution (ESOE2007) at ISWC/ASWC2007, Busan, South Korea*, 2007, pp. 108–120.

[228] Z. Yu, X. Zhou, D. Zhang, C.-Y. Chin, X. Wang, and J. Men, "Supporting context-aware media recommendations for smart phones," *Pervasive Computing*, vol. 5, no. 3, pp. 68–75, 2006. [Online]. Available: ./papers/yu06.pdf

[229] V. Zacharias and S. Braun, "Soboleo - social bookmarking and lightweight ontology engineering," in *Proc. of the WWW'07 Workshop on CKC*. CEUR-WS vol. 273, 2007.

[230] M. Zanker, "A collaborative constraint-based meta-level recommender," in *RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems*. ACM Press New York, NY, USA, 2008, pp. 139–146.

[231] J. Zhang and P. Pu, "A comparative study of compound critique generation in conversational recommender systems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4018 NCS. Springer Verlag, Heidelberg, D-69121, Germany, 2006, pp. 234–243.

[232] L. Zhang, X. Wu, and Y. Yu, "Emergent semantics from folksonomies: A quantitative study," *Journal on Data Semantics*, pp. 168–186, 2006. [Online]. Available: http://dx.doi.org/10.1007/11803034_8

[233] S. Zhang, W. Wang, J. Ford, F. Makedon, and J. Pearlman, "Using singular value decomposition approximation for collaborative filtering," in *CEC '05: Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*. IEEE Computer Society, 2005, pp. 257–264.

[234] N. Zhao, F. Fang, and L. Fan, “An ontology-based model for tags mapping and management,” in *CSSE ’08: Proceedings of the 2008 International Conference on Computer Science and Software Engineering*. IEEE Computer Society, 2008, pp. 483–486.

[235] M. Zhou, S. Bao, X. Wu, and Y. Yu, “An unsupervised model for exploring hierarchical semantics from social annotations,” in *Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea*, ser. LNCS, vol. 4825. Berlin, Heidelberg: Springer Verlag, November 2007, pp. 673–686.

List of Figures

List of Tables